INTEGRATION, the VLSI journal 46 (2013) 382–391

Contents lists available at SciVerse ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Timing–driven variation–aware synthesis of hybrid mesh/tree clock distribution networks ☆

Ameer Abdelhadi [a,*], Ran Ginosar [a], Avinoam Kolodny [a], Eby G. Friedman [b]

[a] Department of Electrical Engineering, Technion–Israel Institute of Technology, Haifa 32000, Israel
[b] Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627, USA

ARTICLE INFO

Article history:
Received 24 October 2011
Received in revised form
10 December 2012
Accepted 11 December 2012
Available online 11 January 2013

Keywords:
Clock distribution networks
Design automation
Integrated circuit design
Low power design

ABSTRACT

Clock skew variations adversely affect timing margins, limiting performance, reducing yield, and may also lead to functional faults. Non-tree clock distribution networks, such as meshes and crosslinks, are employed to reduce skew and also to mitigate skew variations. These networks, however, increase the dissipated power while consuming significant metal resources. Several methods have been proposed to trade off power and wires to reduce skew. In this paper, an efficient algorithm is presented to reduce clock skew variations while minimizing power dissipation and metal area overhead. With a combination of nonuniform meshes and unbuffered trees (UBT), a variation-tolerant hybrid clock distribution network is produced. Clock skew variations are selectively reduced based on circuit timing information generated by static timing analysis (STA). The skew variation reduction procedure is prioritized for critical timing paths, since these paths are more sensitive to skew variations. A framework for skew variation management is proposed. The algorithm has been implemented in a standard 65 nm cell library using standard EDA tools, and tested on several benchmark circuits. As compared to other nonuniform mesh construction methods that do not support managed skew tolerance, experimental results exhibit a 41% average reduction in metal area and a 43% average reduction in power dissipation. As compared to other methods that employ skew tolerance management techniques but do not use a hybrid clock topology, an 8% average reduction in metal area and a 9% average reduction in power dissipation are achieved.

1. Introduction

Non-tree based clock distribution topologies (*e.g.*, clock meshes) exhibit useful topological characteristics due to multi-path signal propagation created by routing redundancies [1–11]. These non-tree clock distribution networks distribute the global clock signal over an integrated circuit, exhibiting high immunity to process, voltage, and temperature (PVT) variations, while tolerating non-uniform switching and an unbalanced distribution of the clocked elements. These networks achieve low and deterministic skew, low skew variations, and low jitter. Clock meshes also overcome late design changes while satisfying tight time-to-market deadlines [1]. Clock meshes constitute an effective alternative for distributing global clock signals, and are used in high performance microprocessors [1] such as the Power4 [2], Digital Alpha [3], Intel® Pentium® 4 [4], and Xeon® [5].

Nevertheless, non-tree clock distribution networks suffer certain drawbacks. These networks are composed of a large number of mesh nodes and unbalanced loads, making these networks difficult to analyze, optimize, and automate [6,12,13]. Routing redundancies require significant resources as compared to optimized tree-based clock distribution networks where point-to-point routing is used [7]. Meshes dissipate higher power [12] due to the large capacitance incurred by the additional metal wires and drivers. Furthermore, clock gating is impractical in most mesh structures. Due to delay differences in the drivers, short-circuit current loops are generated across redundant mesh paths [12]. Increasing process variations [14,15] dissipate more power, since a more tolerant mesh structure dissipates higher power due to greater use of metal and driver oversizing [15]. Several proposals for optimizing non-tree distribution networks have been presented, employing either customized meshes [1–7] or automating the process of adding crosslinks to the clock tree to enhance tolerance and lower power [8,9]. Yet other methods propose removing some edges from a mesh to reduce power while minimally increasing the skew [10,11,22].

☆ This work was supported in part by the Technion Advanced Circuits Research Center (ACRC) and by the Advanced Low Power High End Applications (ALPHA) research consortium.

* Corresponding author.
E-mail addresses: ameer.abdelhadi@gmail.com, ameer@ece.ubc.ca (A. Abdelhadi).

0167-9260/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.vlsi.2012.12.001

While most of these papers focus on skew variations, the approach proposed in this paper manages skew tolerance based on the criticality of the timing margins. The clock signals driving a critical logic path are required to be more tolerant to skew variations to reduce the effect of skew on timing margins and cycle time. Those clock signals that drive a non-critical logic path however must satisfy certain skew variations without affecting the cycle time [16]. By relaxing skew variation requirements in the non-critical paths, power savings can be achieved [1]. The proposed method employs graph-theoretic and geometric algorithms with quasi-linear run time. Using static timing analysis (STA), a physical floorplan, and process information, a hybrid topology of non-uniform clock mesh and unbuffered trees which tolerates clock skew based on timing path criticality is generated. Unbuffered clock trees are used as an extension to the clock mesh within those regions with weak variational requirements. Since process variations and power supply noise in clock buffers are the dominant sources of skew variations [14], unbuffered clock trees, which are comprised primarily of wires and no transistors, exhibit lower skew variations. The clock distribution network is adapted to satisfy clock skew variations while minimizing power dissipation and metal area overhead. The proposed flow has been successfully tested on several benchmark circuits.

While previous work has considered only mesh based clock distribution networks [21], the current work extends and refines the previous skew tolerance management method and introduces a hybrid non-uniform mesh and unbuffered trees clock distribution topology. The solution flow and the algorithms have been adapted to support the hybrid mesh/tree structure while satisfying the skew variation requirements. A statistical distribution of the logic paths shows that the proposed method exhibits a tighter skew requirement bound than the previous method. Hence, design skew variations are closer to the required value, preventing overdesign and reducing metal consumption and power dissipation. Relative to the previous method, an 8% average reduction in metal consumption and a 9% average reduction in power dissipation are achieved.

The rest of the paper is organized as follows. Non-tree clock distribution networks, skew constraints, constraint graphs, and clock skew uncertainty are reviewed in Section 2. The motivation behind this work and a review of previous work on clock mesh synthesis and optimization are discussed in Section 3. The timing–driven variation–aware synthesis of non-uniform mesh and unbuffered tree-based clock distribution networks and the proposed solution are presented along with a run time example in Section 4. The experimental method and results are described in Section 5. Finally, this paper is concluded in Section 6 and future research directions are suggested.

2. Clock meshes and skew

In this section, essential preliminaries are outlined. Specifically, in Section 2.1, non-tree clock topologies, and in particular, clock meshes are described. In Section 2.2, the concept of clock skew, skew uncertainty modeling and notation, and constraint graphs for representing synchronous circuits are reviewed.

2.1. Clock meshes

Clock tree topologies provide a single path for each sink. For distant sinks, these paths are largely separate from each other. Each separate path suffers from delay uncertainty, resulting in skew uncertainty between two sinks. One approach to reduce variations is simply connecting nodes with a crosslink. Hence, the connected nodes will have more than one path from the clock source, mutually compensating each path. This non-tree approach has been manually applied to the Pentium® 4 microprocessor [4], where spines connect multi-clock nodes, as illustrated in Fig. 1(a). Automation of this method has been evaluated in several papers [8].

Since spines or crosslinks connect pairs of nodes and do not cover an entire floorplan (see Fig. 1(b)), metal grids driven by a top level clock driver which span several regions have been introduced. A mesh is a grid of horizontal and vertical metal wire

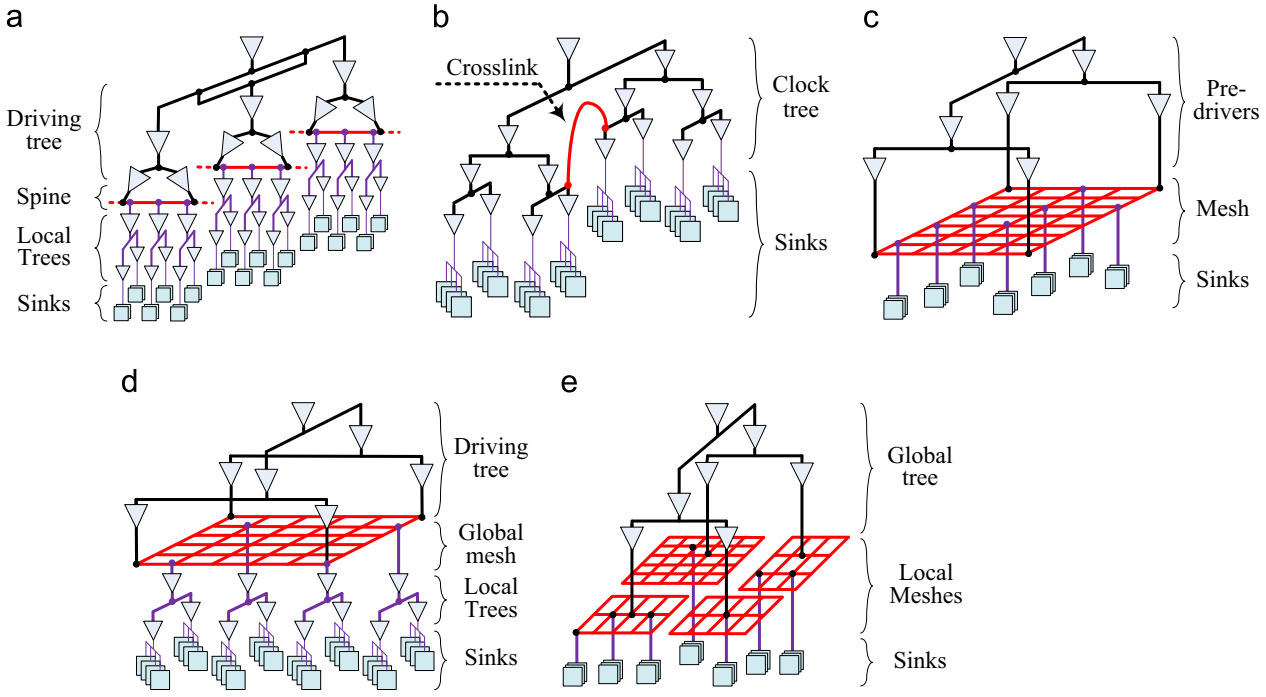

Fig. 1. Non-tree and mesh clock architectures: (a) Pentium® 4 spine, (b) tree with crosslinks, (c) leaf level global mesh, (d) global mesh with local trees (MLT), and (e) global tree with local meshes (TLM).

segments composed of interconnected mesh nodes. Typical mesh topologies consist of three parts: the mesh itself (usually uniform), an upper driving tree, and local interconnects connecting the clock sinks to the mesh, as shown in Fig. 1(c). A wide variety of mesh structures has been proposed. Non-uniform meshes [5,10,11,22] have been developed to save wire resources and power. Design automation and optimization of metal and power versus tolerance tradeoffs are discussed in [10,11,22]. Mesh architectures differ in the locality of the mesh. A global mesh with local trees (MLT) [6] is a mesh fed from a global clock source with local trees distributing the clock to local regions (Fig. 1(d)), while a global tree with local meshes (TLM) [6] is a clock tree fed from a global clock source with a local mesh at each leaf (Fig. 1(e)). Other hybrid mesh-style structures are also possible.

2.2. Skew constraints and uncertainty

Synchronous circuits comprise data paths, where each combinational logic path is located between two registers. A clock network connects the clock source to a collection of clock sinks $S=\{s_1,s_2,\ldots,s_n\}$. Two registers are *sequentially-adjacent* if the registers are connected with a combinational data path [1], as illustrated in Fig. 2. The maximum permissible delay for a local data path bounded by two sequentially-adjacent registers is

$$P^{i,j}_{\text{Delay,max}} = T_{\text{clock}} - t_{\text{setup}} - skew^{i,j}_{\text{max}}, \tag{1}$$

where T_{clock} denotes the clock period, t_{setup} is the setup time of the bounding registers, and $skew^{i,j}_{\text{max}}$ denotes the maximum clock skew between two bounding registers [1]. The maximum clock skew $skew^{i,j}_{\text{max}}$ is the difference in the arrival time between two sequentially-adjacent registers. If d_i and d_j are the delay (maximum or minimum) of the clock signals arriving at the registers i and j, respectively, the skew between two adjacent registers is

$$skew^{i,j}_{\text{max}} = \max_{i,j}(|d_i - d_j|). \tag{2}$$

The clock skew is therefore bounded by the following *maximum skew constraint*,

$$skew_{i,j} \le T_{\text{clock}} - T_{\text{setup}} - P^{i,j}_{\text{Delay,max}}. \tag{3}$$

2.2.1. Clock skew uncertainty

As technology scales, the effect of PVT variations on clock skew is aggravated [14,15]. Clock skew can be modeled as composed of both deterministic and probabilistic elements [13,18]. The following notation is employed to describe these probabilistic effects. For two sequentially-adjacent registers i and j, the deterministic nominal skew components is $skew^{i,j}_{\text{nom}}$, and the maximum skew variation of the probabilistic component is $\delta^{i,j}_{\text{max}}$. The mean of the skew between two sequentially-adjacent registers i and j is denoted by $\mu^{i,j}_{\text{skew}}$ and the standard deviation is denoted by $\sigma^{i,j}_{\text{skew}}$. A possible *3σ design target* may require that, for instance, the maximum skew will be limited by

$$skew^{i,j}_{\text{max}} = \mu^{i,j}_{\text{skew}} + 3\sigma^{i,j}_{\text{skew}}. \tag{4}$$

2.2.2. Constraint graph

Synchronous circuits are represented as a directed multi-graph G_C [1,16,17]. Each clock sink is represented by a vertex $v_i \in G^V_C$, such that $G^V_C = S$. Each local data path located between two sequentially-adjacent clock sinks i and j is represented by a weighted edge $e_{i,j} \in G^E_C$ connecting the two vertices v_i and v_j (see Fig. 3). The graph edges are therefore

$$G^E_C = \{e_{i,j} = v_i \sim v_j \,|\, P^{i,j}_{\text{Delay}} < \infty; v_i, v_j \in G^V_C\}. \tag{5}$$

The edges can be weighted by any corresponding combinational data path property, such as delay, margin, and skew. Besides the edge weights, attributes can be attached to either edges or vertices. For vertices, any corresponding sink attribute can be used, such as the sink capacitance, location, clock delay, and data arrival time.

3. Hybrid nonuniform mesh/tree synthesis: Motivation and related work

Motivation for using timing information as a criterion for clock mesh synthesis is discussed in Section 3.1. A review of previous clock mesh synthesis methods is presented in Section 3.2.

3.1. Motivation

While some approaches have been proposed to minimize clock skew, these methods usually incur an increase in power consumption. Other methods exploit useful skew by scheduling clock skew to increase the maximum frequency [17]. These methods, however, suffer from increased clock skew uncertainty with process scaling. This issue limits circuit performance since timing margins are reduced [14,15].

Timing margins are provided to overcome uncertainty in the clock arrival time caused by within-die variations. Poor timing margins, namely, critical paths within a digital circuit are sensitive to skew variations. Hence, skew variations should be minimized, particularly in those clock paths that drive the critical paths. This method, however, is usually achieved at the expense of higher power dissipation [1]. This paper focuses on differential treatment of the clock and selective management of the skew.

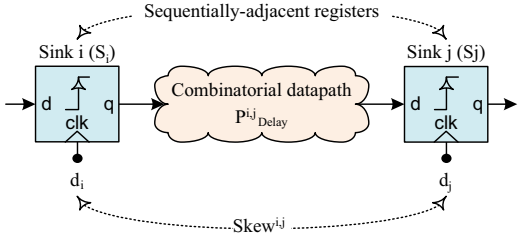

Fig. 2. Sequentially-adjacent registers s_i and s_j with $skew^{i,j}$ clock skew, bounding a combinational data path with propagation delay $P^{i,j}_{\text{Delay}}$.

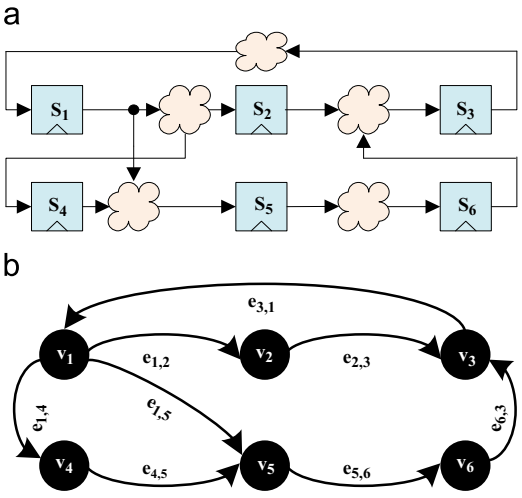

Fig. 3. Mapping of constraint graph: (a) synchronous circuit and (b) corresponding constraint graph.

The more critical a path, the more sensitive the path is to skew variations and the greater the effort to reduce these skew variations [16]. Paths that are non-critical are assigned a lower effort to reduce skew variations. Path criticality prioritization is intended to save power since skew variations are not minimized on those paths that do not affect circuit operation or speed. This approach is in contrast to skew optimization methods that aim to reduce maximum or nominal skew over an entire circuit.

3.2. Related work

The effect of increasing process variations is particularly pronounced in clock distribution networks, since skew variations strongly influence system performance and require careful analysis of minimum delays [14,15]. Non-tree clock meshes, although useful in mitigating process variations, are difficult to analyze and automate due to the complex structure. Most mesh clock networks are manually designed in high performance applications such as microprocessors [1–7]. Several approaches automate the clock mesh design process. Mesh sizing and, in particular, segment wire width sizing using network flow algorithms have been used to optimize nominal skew rather than skew variations [19]. Other methods start from a clock tree and incrementally add crosslinks among the tree nodes or leaves. Crosslinks are inserted between those nodes exhibiting significant variations. The objective is to add the fewest number of crosslinks while reducing the maximum or average variations [8,9]. Other approaches start with a fully uniform mesh, identify and remove redundant segments whose effect on variations is minimal by applying network theory techniques, thereby trading off variations with wire length and power. A set-cover problem is solved to obtain mesh-based predriver minimum buffers [10,11,22]. The initial uniform grid is designed to ensure that metal redundancies within the grid satisfy target skew requirement [10,22]. Whereas some methods generate a clock distribution network with nonuniform meshes, adapting the mesh structure to satisfy skew variation requirements [21], other methods combine meshes and trees to construct a hybrid topology [23,24]. The objective of these hybrid methods is to reduce the maximum nominal skew of the clock tree by adding a mesh at the leaves of the tree [23]. Other hybrid topology methods combine meshes with zero skew trees (ZST) to construct a zero skew clock distribution network [24]. Nevertheless, the above methods do not exploit circuit timing information for selectively reduction of skew variation.

4. Synthesis of hybrid mesh/tree clock distribution networks

Timing–driven variation–aware hybrid mesh/tree clock distribution network synthesis is discussed in this section. The problem formulation and solution approach are presented in Sub Sections 4.1 and 4.2, respectively.

4.1. Problem statement

The problem of managing skew variations can be formulated using the notations defined in Section 2.2.

4.1.1. Inputs

Given a circuit connectivity, physical placement, and static timing analysis, including (1) a set of clock sinks in the Manhattan plane $S=\{s_1,s_2,\ldots,s_n\}$, (2) maximum skew constraints between each set of sequentially-adjacent registers s_i and s_j, namely, the maximum permissible skew $skew^{i,j}_{allowed}$, and (3) the relative tolerance parameter ξ. The relative tolerance parameter ξ is a user defined parameter denoting the upper bound of the maximum skew variation ratio over all maximum skew constraints allowed for all data paths

$$\xi \geq \frac{\delta^{i,j}_{\max}}{skew^{i,j}_{\text{allowed}}} \quad \left(\forall e_{i,j} \in G^E_C\right). \tag{6}$$

4.1.2. Problem formulation

Construct a hybrid mesh/tree clock distribution network consisting of nonuniform meshes and unbufferd trees with reduced wire length and power consumption. This clock distribution network limits the fraction of the maximum skew variation over all maximum skew constraints by ξ for every combinational data path, as expressed by (6).

The mesh density is the number of nodes connecting the wire segments within the mesh. A uniform mesh consists of m horizontal segments and n vertical segment requiring $n \times m$ nodes.

Lower levels of ξ lead to further reductions in clock skew variations at the expense of a denser mesh, longer wire length, and higher power consumption. Alternatively, higher levels of ξ lead to a sparse mesh or even a tree, since skew variation reduction requirements are relaxed. Thus, power and metal savings could be achieved. ξ may be tuned by considering the tradeoff among power dissipation, metal consumption, and design robustness.

4.2. Hybrid mesh/tree construction algorithm

The algorithm places multiple clock meshes or unbuffered trees over certain rectangular regions. The meshes may be of different densities. The unbuffered trees are attached to the nonuniform mesh structure as extensions of this mesh. The regions may partly overlap. Each region covered by a mesh is associated with a specific maximum skew constraint, which determines the density of the corresponding mesh. The algorithm comprises four phases. In phase I, the constraint graph is derived from static timing analysis and connectivity information. A skew map, comprising floorplan regions with different skew variation requirements, is extracted in phase II. The overlap among the skew regions is removed in phase III. Finally, a mesh with a specific density or an unbuffered tree is matched to each skew region in phase IV, constructing a hybrid mesh/tree topology.

4.2.1. Phase I: Derivation of constraint graph

A constraint graph, as defined in Section 2.2 above, is derived from the circuit connectivity information. See the example shown in Fig. 4. The vertices represent clock sinks and the edges represent data paths between vertices. The edge weight $w(e_{i,j})$ represents the maximum skew constraint of the local data path represented by the edge $e_{i,j}$, $w(e_{i,j})=skew^{i,j}_{\text{allowed}}$. Each vertex can be assigned multiple attributes, such as the capacitance of the corresponding sink $C(v_i)=\text{Capacitance}(s_i)$ and the geometric location of the clock sinks of the vertex.

As the algorithm progresses, some vertices are merged, representing a geometric rectangular region containing multiple clock sinks. As the vertices are merged, the inner connectivity between the constituent sinks is ignored, and only the inter-vertex connectivity is preserved. The attributes of a vertex represent the properties of all sinks included in the vertex: the capacitance is the sum of all sink capacitances and the geometric location is replaced by a rectangular bounding box covering the physical location of all of the corresponding clock sinks.

Misplaced registers, *e.g.*, sequentially-adjacent registers bounding a critical timing path and placed diagonally apart, may cause undesired results. These cases are reported at an early stage,

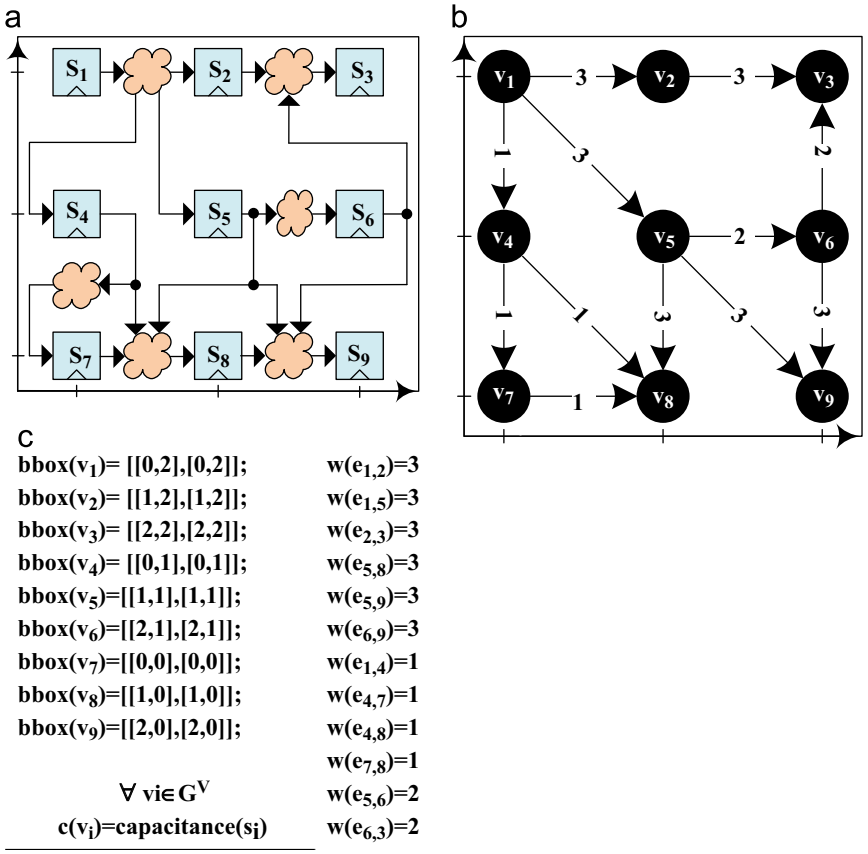

Fig. 4. Phase I, constraint graph construction: (a) synchronous circuit floorplan and connectivity with placed registers, (b) corresponding constraint graph; edge weights are the maximum allowed skew, and (c) the vertex attributes and edge weights are the initial values.

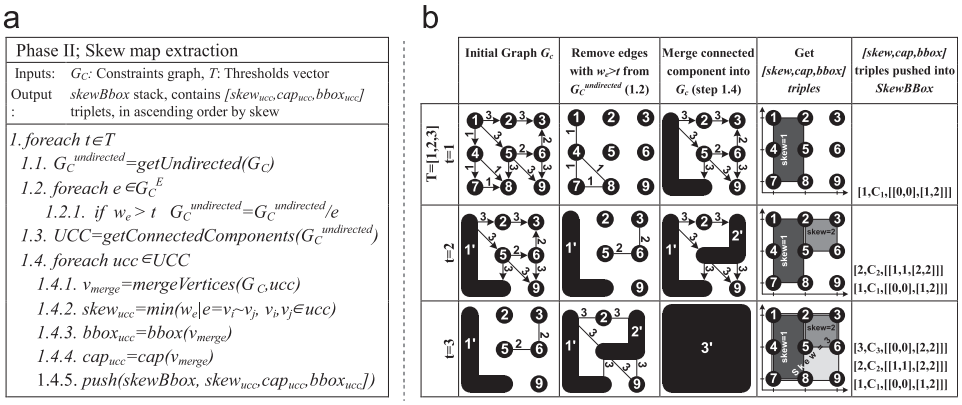

Fig. 5. Phase II algorithm for skew map extraction: (a) algorithm description (mergeVertices() is shown in Fig. 6), and (b) execution example. The rows are iterations and columns are steps of the algorithm.

suggesting replacing interfering registers to avoid unnecessary design iterations.

4.2.2. Phase II: Skew map extraction

The rectangular regions satisfying certain skew requirements are identified. The algorithm iterates over an increasing threshold level. A vector T contains all pre-determined threshold values and defines basic time steps for mesh construction. At each iteration, each connected vertex is interconnected by edges with a weight below the current threshold and merged into a new vertex. A geometric bounding box covering all clock sinks within the same vertex is identified. The algorithm is described in Fig. 5(a). A threshold $t \in T$ is selected, edges with weight higher than the threshold are eliminated (step 1.2), the corresponding vertices are identified as a connected component (step 1.3), and these vertices are merged into one larger vertex (step 1.4.1). The skew constraint of this new vertex is the tightest skew constraint among all edges inside the corresponding connected component (step 1.4.2). The algorithm terminates when no threshold values remain in T, or only one vertex remains.

A run time example of Phase II is shown in Fig. 5(b). The merge operation is performed by the subroutine shown in Fig. 6, as follows: The inner edges inside a connected component are removed and externally connected edges are connected to the new merged vertex (step 2). The attributes of all vertices inside the connected component are merged into the attributes of the corresponding new vertex. The capacitance is the sum of all inner

capacitances (step 3) and the bounding box bounds all inner sinks or inner bounding boxes (step 4). The merge operation is illustrated in Fig. 7. Note that this phase produces a set of possibly overlapping rectangular regions.

Extracting and merging all connected components requires $O(|G_C^V|+|G_C^E|)$. The overall run time complexity of phase II is therefore $O(|T|(|G_C^V|+|G_C^E|))=O(|S|)$.

4.2.3. Phase III: Overlap removal

Partly overlapping rectangular skew regions are merged into polygons, and skew levels are assigned to the polygons. When two skew regions overlap, the tighter skew constraint prevails and is inherited by the resulting polygon, as illustrated in Fig. 8(a). The input [*skew*, *capacitance and bounding box*] triplets are sorted in ascending order of skew. Iteratively, a triplet with a tighter skew constraint is removed from the input stack (step 2.1).

Merge vertices procedure
Inputs: G: Graph, V: Group of vertices to be merged Output: V_{merge}: Merged vertex with merged attributes
MergeVertices(G,V) *1. create new vertex* v_{merge} *2. foreach* $v \in V$ *2.1. foreach* $e=v\sim v'$ *2.1.1. if* $v' \notin V$ $e = v' \sim v_{merge}$ *2.2. remove* v *3.* $C(v_{merge})=\sum_{v\in V} C(v)$ *4.* $Bbox(v_{merge})=[min(x_0),min(y_0),max(x_1),max(y_1)]$ | $(x_0,y_0,x_1,y_1)=bbox(v),\ v\in V$ *5. return* v_{merge}

Fig. 6. Vertex merging algorithm (part of Phase II).

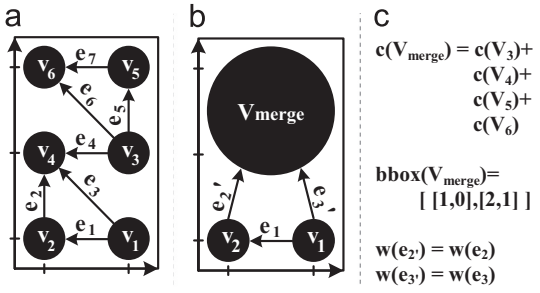

Fig. 7. Merge example: (a) constraint graph, vertices are placed at the same place as the corresponding registers, (b) constraint graph after merge, and (c) value of attributes after merge.

The circuit floorplan is filled with non-overlapping skew regions (steps 2.2 and 2.3). Polygon shaped skew regions are constructed from overlapping regions. A run time example of Phase III is shown in Fig. 8(b).

Polygon union and intersection operations can be performed in $O(n\ log(n))$ steps using a segmented tree data structure [20], where n is the total number of polygon segments. Since $n=O(|G_C^V|)=O(|S|)$, the computational runtime of this phase is therefore quasi-linear, $O(|S|log(|S|))$.

4.2.4. Final phase IV: Hybrid mesh/tree generation

A clock mesh or unbuffered tree is designed for each of the non-overlapping skew polygons. Each mesh or tree should satisfy the skew requirement of the polygon. If the maximum allowed skew for a specific region is $skew_{\text{maxallowed}}$, the maximum allowed variation is $\delta_{\max}^{skew} < \xi \cdot skew_{\text{maxallowed}}$. For each of the non-overlapping skew polygons, an unbuffered tree is synthesized and simulated to derive the required characteristics, *e.g.*, delay variations and slew rate. If the variation target, namely $\delta_{\max}^{skew}$, and the user defined slew rate are satisfied by the unbuffered tree, the leaves of this unbuffered tree will be connected to all of the clock sinks within the current skew region. The root of this unbuffered tree is attached to the closest mesh by the shortest Manhattan path between this attaching point and the center of mass of the clock sinks within the current skew region. The center of mass is the average location of all of the clock sinks, weighted by the sink capacitances [25]. Let V_{UBT} denote a set of vertices representing the clock sinks within the current skew region. The center of mass of these vertices can be expressed as

$$(x_{cm},y_{cm}) = \left(\frac{\sum_{v\in V_{UBT}} c(v)x(v)}{\sum_{v\in V_{UBT}} c(v)}, \frac{\sum_{v\in V_{UBT}} c(v)y(v)}{\sum_{v\in V_{UBT}} c(v)}\right), \quad (7)$$

where (x_{cm},y_{cm}) and $(x(v),y(v))$ denote, respectively, the coordinates in the Manhattan grid of the center of mass, and the clock sink represented by v. $c(v)$ denotes the capacitance of the sink represented by v.

Since unbuffered trees consist of only metal interconnect, the skew variation of these trees is proportional to the interconnect delay variation. Due to slew rate degradation of the clock signal along the unbuffered long wires, those skew regions covered with unbuffered trees are restricted by the area and sink capacitances [26]. If the target skew variation and slew rate are not achieved with unbuffered trees, meshes are used. The skew variation is inversely proportional to the mesh density. The density of each mesh is therefore tuned to match the required skew variation. Skew as a function of mesh density has also been discussed in [10] and [22]. The initial mesh density is bounded using an approximation from [10] which obtains the maximum skew for

a

Phase III; Overlap removal
Inputs: *[skew,capacitance,bbox]* triplets from phase II Output: *skewPolygon* stack, containing *[skew,cap..,polygon]* triplets
1. covered=Ø *2. while skewBbox≠Ø* *2.1. [skew,cap.,bbox]= pop(reversed((skewBbox))* *2.2. polygon=covered^c∩bbox* *2.3. covered=covered∪bbox* *2.4. push(skewPolygon[skew,cap,polygon])*

b

reversed(skewBbox)	skew	bbox	polygon	covered	[skew,cap,polygon]
[1,C1,[[0,0],[1,2]]] [2,C2,[[1,1],[2,2]]] [3,C3,[[0,0],[2,2]]]	1				[1,C1,[[0,0],[0,2],[1,2],[1,0]]]
[2,C2,[[1,1],[2,2]]] [3,C3,[[0,0],[2,2]]]	2				[2,C2,[[1,1],[1,2],[2,2],[2,1]]] [1,C1,[[0,0],[0,2],[1,2],[1,0]]]
[3,C3,[[0,0],[2,2]]]	3				[3,C3,[[1,0],[1,1],[2,1],[2,0]]] [2,C2,[[1,1],[1,2],[2,2],[2,1]]] [1,C1,[[0,0],[0,2],[1,2],[1,0]]]

Fig. 8. Phase III; generating nonoverlapping skew map (a) algorithm description, and (b) execution example. The rows are iterations and columns are steps of the algorithm.

a given mesh density. The initial mesh is simulated to achieve a more accurate skew bound. If the skew bound does not meet the requirements, the mesh density is refined and simulated again. Optimized pre-drivers are placed by solving a set-covers problem. The same set-covers approach from [10,11] has been employed.

Johnson's standard greedy algorithm is employed to solve the set-covers problem [27]. At each step of Johnson's algorithm the subset that covers the largest number of the remaining uncovered elements is picked up, until all elements are covered. The complexity of the greedy algorithm is $O(|S|\ |B|)$, where S is the clock sinks set and B is the group of all possible pre-driver buffers from a predefined buffer locations and library of sizes. The set-cover database is maintained by a priority queue; hence the execution run time can be achieved in $O(\sum_{b\in B}|S_b|)$, where S_b is the group of clock sinks that can be covered by the buffer b. On the other hand $O(\sum_{b\in B}|S_b|)$ is equivalent to $O(\sum_{s\in S}|B_s|)$, where B_s is the group of buffers that can drive the sink s. Since each clock sink can be driven by a finite set of neighboring buffers, $|B_s|$ is constant, and the overall run time complexity of set-covers standard greedy algorithm is $O(|S|)$, namely linear in the number of clock sinks.

The overall run time for the entire algorithm is quasi-linear in the number of clock sinks: $O(|S|\cdot log(|S|))$. An example output of the algorithm is illustrated in Fig. 9.

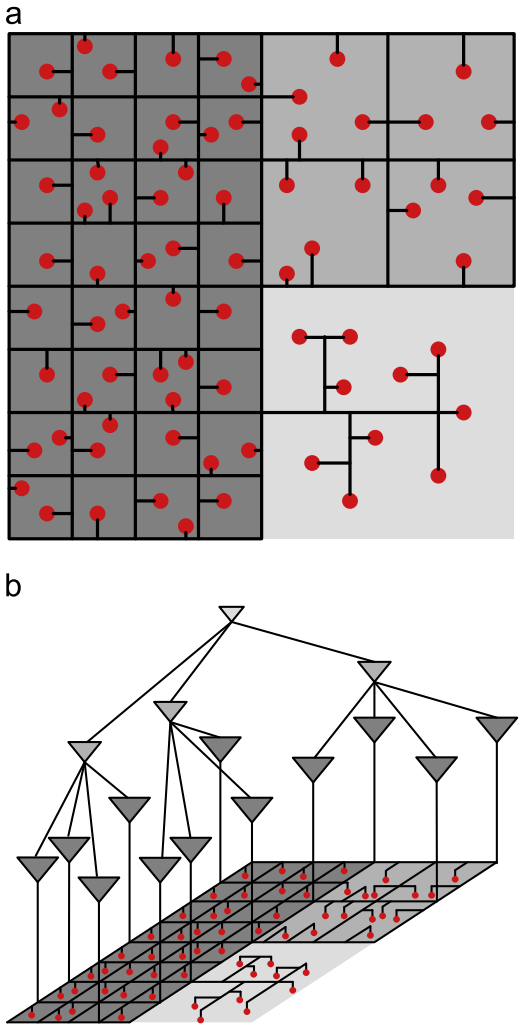

Fig. 9. Example of the proposed hybrid nonuniform mesh/unbuffered tree structure; (a) The skew map is shown in the background. Darker regions indicate a tighter variation target. The circular spots are clock sinks, and (b) a 3-D representation of the network including the driving tree.

5. Experimental results

To verify the capability of the proposed method to reduce power and wire length consumed by the mesh architecture, and to compare these capabilities to previous methods, several experiments have been conducted. These experiments are described and the results are discussed in Sections 5.1 and 5.2, respectively.

5.1. Flow and design environment

The proposed algorithm has been implemented in Perl and TCL. Experiments have been performed on several circuits from the ISCAS89 sequential benchmark suite. These benchmark circuits have been designed using the Virage Logic SiWare™ standard cell logic library with a 65 nm process operating at a 1 GHz frequency. An RTL representation of the benchmark circuits has been synthesized into a netlist using Synopsys® Design Compiler® Ultra (DC Ultra) and placed and routed by the Cadence® SoC Encounter™ RTL-to-GDSII System. A TCL hook procedure is used to construct the constraint graphs, which are imported into an XML database. The proposed algorithm generates mesh and pre-driver locations. Unbuffered clock trees are routed with the Cadence® SoC Encounter™ internal clock router. The Cadence® SoC Encounter™ constructs the physical layout. Results are analyzed using the Cadence® Virtuoso® UltraSim Full-Chip Simulator, a transistor-level FastSPICE circuit simulator. A statistical distribution of the logic paths is also obtained from Cadence® Virtuoso® UltraSim advanced Monte Carlo statistical analysis.

5.2. Results

The results of applying the proposed algorithm to several benchmark circuits are listed in Table 1. The first five rows list, respectively, the benchmark name, number of clock sinks, logic gates, logic paths, and skew regions. The following two row blocks list, respectively, the total wire length and power consumption. The experimental parameter ξ is varied around a typical value. The granularity of the skew regions is affected by the circuit characteristics and the user defined threshold vector T. The distance between subsequent threshold values controls the resolution of the generated skew map. As the distance between threshold values decreases, the skew regions become finer; however, the algorithm run time increases due to additional mesh construction loops. Hence, the threshold vector controls a design quality versus run time tradeoff. The nominal slew rate is typically 10% of the clock period [2]; hence the clock slew was constrained to 100 ps. The experiment evaluates the relationship among the metal resources, power consumption, and relative skew tolerance parameter ξ. As expected, a larger ξ increases the wire length and power consumption but reduces the maximum skew (see Fig. 10(d)). Comparing these results to the methods proposed in [10,11,], and [21], a typical value of $\xi=1$ improves the wire length and power dissipation, as depicted in Fig. 10. As compared to other nonuniform mesh construction methods without skew tolerance managing capabilities [10,11], the proposed method achieves a 41% average reduction in metal consumption and a 43% average reduction in power dissipation. Relative to other methods that employ a skew tolerance managing mechanism but do not use a hybrid clock topology [21], an 8% average reduction in metal consumption and a 9% average reduction in power dissipation is achieved.

A statistical distribution of the logic path delays based on the proposed method is compared to [21] and listed in Table 2. A histogram of the logic path distribution is plotted in Fig. 11.

Table 1
Results of the proposed algorithm as compared with other approaches [10,11,21].

Test case	s9234	s5378	s13207	s15850	s38584	s35932	Average
Clock sinks	135	165	500	566	1426	1728	753
Logic gates	5.6K	2.8K	8K	9.8K	19.2K	16K	10.2K
Logic paths	3.1K	5.4K	12.4K	18.6K	54.5K	43.3K	22.9K
Skew regions	3	3	4	6	8	6	5
Wire Length (μm)							
$\xi=1.1$	12,113	19,436	46,533	40,379	156,197	212,638	81,216
$\xi=1.0$	11,713	16,683	47,623	46,139	151,995	218,274	82,071.2
$\xi=0.9$	12,506	19,725	46,893	42,046	161,538	223,442	84,358.3
[10]	33,610	31,009	82,884	84,055	256,567	349,432	139,593
[11]	27,177	24,911	109,538	100,778	262,528	321,293	141,038
[21]	13,376	20,839	51,443	45,628	166,274	239,342	89,483.7
Power (mw)							
$\xi=1.1$	4.85	4.12	10.33	11.37	36.23	38.74	17.6
$\xi=1.0$	5.18	4.53	11.71	12.55	37.46	39.81	18.5
$\xi=0.9$	6.7	4.66	12.04	13.53	39.17	42.39	19.7
[10]	8	6.7	20.6	22	65.2	73.5	32.7
[11]	6.7	6.72	23.8	23.8	60.9	74.3	32.7
[21]	5.27	4.84	12.59	13.93	41.18	44.25	20.3

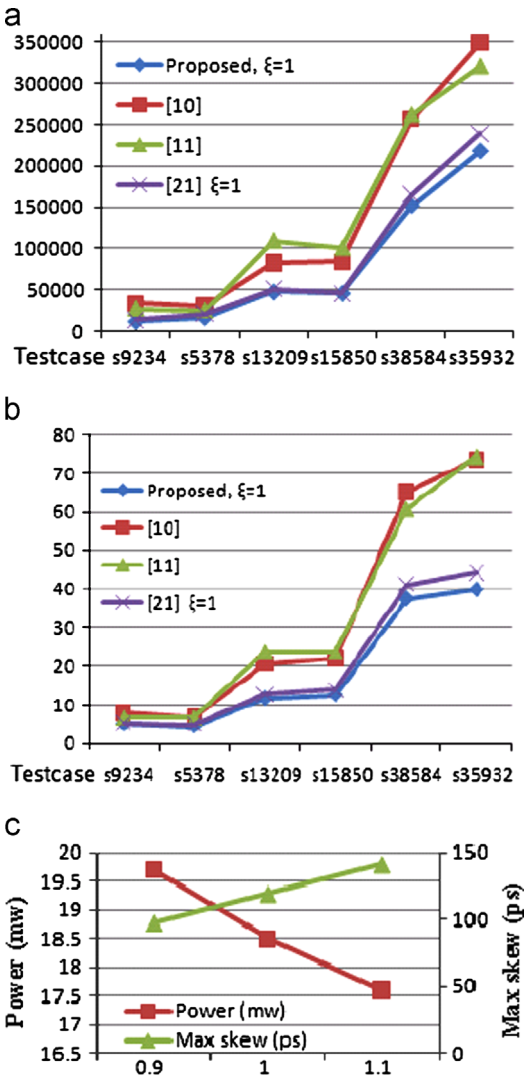

Fig. 10. Proposed method with $\xi=1$ vs. [10,11,21]: (a) wire length, (b) power, and (c) power and maximum skew vs. relative skew tolerance parameter ξ.

As shown in this figure, both the proposed method and [21] successfully bound the maximum skew variation ratio over all maximum allowed skew constraints by ξ. Note that the proposed method shifts the logic path delays toward the bounding value (ξ), exhibiting a tighter bound than [21]. A tighter bound means that the design skew variation is closer to the target value; hence, reducing the required metal resources and power dissipation. Hybrid clock topologies therefore can successfully achieve design targets, while avoiding overdesign.

A statistical distribution of the logic paths is not available for [10] or [11]; hence, a comparison of the distribution of the logic path delays is not possible. While [10] and [11] aim to reduce the overall maximum skew, thereby incurring an overdesign overhead, the proposed method selectively reduces skew variations to avoid overdesign, thereby saving metal and power.

The CPU run time of the entire design flow is considerably low. The largest test case with 1728 clock sinks executes in less than 40 s on a Linux machine with a 4GB of RAM and a 2 GHz Intel based processor.

6. Conclusions

An efficient graph-theoretic and geometric quasi-linear algorithm for managing clock skew tolerance is presented in this paper. Skew variations are managed while considering the criticality of the timing of each data path. An algorithm and flow for planning and synthesizing non-uniform clock meshes are integrated with current CAD tools and demonstrated on a 65 nm CMOS process and cell library. Experimental results on a set of benchmark circuits exhibit a 41% average decrease in metal wire length and a 43% average decrease in power dissipation as compared to existing methods that do not employ selective reduction of skew variations. Comparison to other methods with variation managing techniques using only meshes exhibits an 8% average reduction in metal consumption and a 9% average reduction in power dissipation.

These results demonstrate that managing skew tolerance by wisely prioritizing critical paths saves significant metal resources and dissipates less power as compared to traditional methods. Furthermore, hybrid clock structures consisting of both nonuniform

Table 2
Number of logic paths with respect to the maximum skew variation ratio over all maximum allowed skew constraints ($\xi=1$).

$\delta_{max}/skew_{allowed}$	s9234		s5378		s13207		s15850		s38584		s35932		Average	
	Proposed	[21]	Proposed	[21]	Proposed	[21]	Proposed	[21]	Proposed	[21]	Proposed	[21]	Proposed	[21]
0–0.05	65	9	2	5	4	25	11	56	0	311	0	173	14	97
0.05–0.1	23	16	4	16	9	74	24	149	0	654	43	346	17	209
0.1–0.15	1	28	14	27	22	223	48	249	71	981	74	606	38	352
0.15–0.2	5	37	9	38	42	248	22	260	55	1254	178	996	52	472
0.2–0.25	29	40	18	49	32	273	86	335	327	1363	260	1905	125	661
0.25–0.3	26	99	39	70	154	298	134	428	436	1690	346	1602	189	698
0.3–0.35	34	53	32	92	58	310	260	484	382	1799	173	1386	157	687
0.35–0.4	159	65	47	130	72	335	223	632	491	2017	476	2078	245	876
0.4–0.45	102	68	49	173	91	360	195	651	600	2126	563	1949	267	888
0.45–0.5	67	84	50	157	105	372	210	484	763	2344	909	1797	351	873
0.5–0.55	29	43	72	184	25	397	279	502	981	2562	779	2208	361	983
0.55–0.6	108	121	92	340	267	446	428	577	1363	2627	996	2295	542	1068
0.6–0.65	290	174	363	281	880	632	502	688	1744	2671	1126	1819	818	1044
0.65–0.7	215	229	297	340	769	546	744	800	1690	2834	953	2122	778	1145
0.7–0.75	233	189	527	416	546	632	818	688	1962	2889	1559	2641	941	1243
0.75–0.8	316	301	585	524	1029	781	1693	1395	3924	3543	1819	2382	1561	1488
0.8–0.85	483	347	767	502	1562	1141	2734	1637	7903	4197	2815	2901	2711	1788
0.85–0.9	357	456	628	437	2418	1916	4073	2399	12917	4633	5265	3551	4276	2232
0.9–0.95	301	381	767	902	2282	1699	2864	3032	10028	7848	14159	5932	5067	3299
0.95–1	223	282	994	783	1897	1401	3013	2548	7935	5668	9799	4243	3977	2488

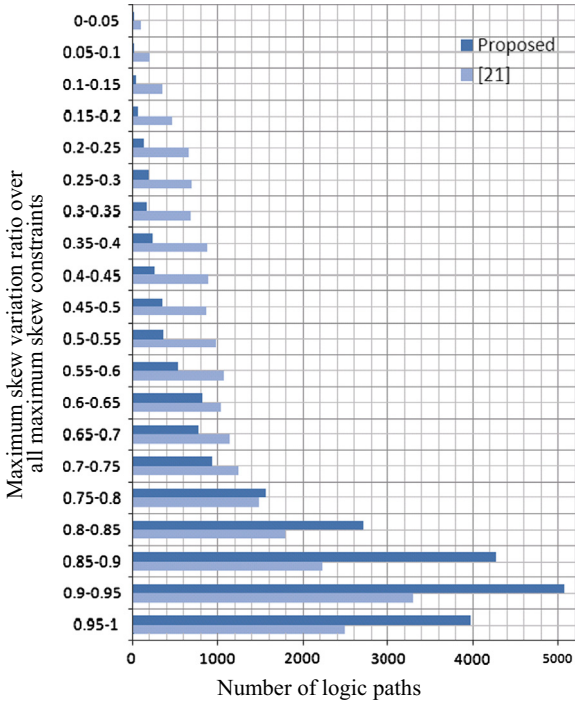

Fig. 11. Distribution of the logic paths of the proposed method, compared to [21], showing the maximum skew variation ratio over all maximum allowed skew constraints.

meshes and trees are capable of further reducing power and metal consumption.

Future improvements of the algorithm are possible. Other design parameters such as the metal width and layer could be integrated into the optimization process. Since constraint graph extraction is computationally expensive, registers at the same local region could be clustered into one node before extraction. The algorithm presented here can be adapted to automate the crosslink insertion process [8,9]. Rather than inserting crosslinks to reduce maximum variations, the crosslinks could be inserted according to the criticality of the individual data paths. The algorithm presented here targets a mesh for zero skew; useful skew [17] may also be considered. The selective resource management idea could be adapted to construct and optimize power grid networks in addition to clock distribution networks. A power grid is constructed to satisfy current consumption for each consumer. This method reduces metal area, hence power grid capacitance. However, some applications require increased capacitance within the power grid for mitigating power supply noise, so that this method should be employed with care.

Acknowledgment

The authors would like to thank the VLSI Systems Research Center at the Technion for providing CAD tools, and Virage Logic Corpopration for providing technology process and cell library information.

References

[1] E.G. Friedman, Clock distribution networks in synchronous digital integrated circuits, Proceedings of the IEEE 89 (5) (2001) 665–692.
[2] P.J. Restle et al., The clock distribution of the power4 microprocessor, in: Proceedings of the IEEE International Solid-State Circuits Conference, February 2002, pp. 1.144–1.145.
[3] T. Xanthopoulos et al., The design and analysis of the clock distribution network for a 1.2 GHz alpha microprocessor, in: Proceedings of the IEEE International Solid-State Circuits Conference, February 2001, pp. 402–403.
[4] N.A. Kurd, J.S. Barkatullah, R.O. Dizon, T.D. Fletcher, P.D. Madland, A multi-gigahertz clocking scheme for the Pentium® 4 microprocessor, IEEE Journal of Solid-State Circuits 36 (11) (2001) 1647–1653.
[5] S. Tam et al., Clock generation and distribution of a dual-core xeon processor with 16MB L3 Cache, in: Proceedings of the IEEE International Solid-State Circuits Conference, February 2006, pp. 1512–1521.
[6] C. Yeh et al., Clock distribution architectures: a comparative study, in: Proceedings of the IEEE International Symposium on Quality Electronic Design, March 2006, pp. 85–91.
[7] P.J. Restle, A. Deutsch, Designing the best clock distribution network, in: Proceedings of the IEEE Symposium on VLSI Circuits, June 1998, pp. 2–5.
[8] A. Rajaram, J. Hu, R. Mahapatra, Reducing clock skew variability via cross-links, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25 (6) (2006) 1176–1182.
[9] I. Vaisband, R. Ginosar, A. Kolodny, E.G. Friedman, Power efficient tree-based crosslinks for skew reduction, in: Proceedings of the ACM/IEEE Great Lakes Symposium on VLSI, May 2009, pp. 285–290.
[10] A. Rajaram, D.Z. Pan, Meshworks: an Efficient framework for planning, synthesis and optimization of clock mesh networks, in: Proceedings of the

IEEE Asia and South Pacific Design Automation Conference, January 2008, pp. 250–257.
[11] G. Venkataraman, Z. Feng, J. Hu, P. Li Combinational algorithms for fast clock mesh optimization, in: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, November 2006, pp. 563–567.
[12] A.L. Sobczyk, A.W. Luczyk, W.A. Pleskacz, Power dissipation in basic global clock distribution networks, Proceedings of the IEEE Design and Diagnostics of Electronic Circuits and Systems (2007) 1–4.
[13] G. Tosik, L.M.S. Gallego, Z. Lisik, Different approaches for clock skew analysis in present and future synchronous IC's, in: Proceedings of the International Conference on Computer as a Tool, September 2007, pp. 1227–1232.
[14] V. Mehrotra, D. Boning, Technology scaling impact of variation on clock skew and interconnect delay, in: Proceedings of the IEEE International Interconnect Technology Conference, June 2001, pp. 122–124.
[15] S. Abe, M. Hashimoto, T. Onoye, Clock skew evaluation considering manufacturing variability in mesh-style clock distribution, in: Proceedings of the IEEE International Symposium on Quality Electronic Design, March 2008, pp. 520–525.
[16] D. Velenis, M.C. Papaefthymiou, E.G. Friedman, Reduced delay uncertainty in high performance clock distribution networks, in: Proceedings of the IEEE Design Automation and Test in Europe Conference, March 2003, pp. 68–73.
[17] I.S. Kourtev, B. Taskin, E.G. Friedman, Timing Optimization Through Clock Skew Scheduling, Second Edition, Springer Science+Business Media, 2009.
[18] S.D. Kugelmass and K. Steiglitz, A probabilistic model for clock skew, in: Proceedings of the IEEE International Conference on Systolic Arrays, May 1988, pp. 545–554.
[19] M.P. Desai, R. Cvijetic, J. Jensen, Sizing of clock distribution networks for high performance CPU chips, in: Proceedings of the IEEE/ACM Annual Design Automation Conference, June 1996, pp. 389–394.
[20] F.P. Preparata, M.I. Shamos, Computational Geometry: An Introduction, Springer-Verlag, 1985, pp. 13–15.
[21] A. Abdelhadi, R. Ginosar, A. Kolodny, E.G. Friedman, Timing-driven variation-aware nonuniform clock mesh synthesis, in: Proceedings of the ACM/IEEE Great Lakes Symposium on VLSI, May 2010, pp. 15–20.
[22] A. Rajaram, D.Z. Pan, MeshWorks: A Comprehensive Framework for Optimized Clock Mesh Network Synthesis, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 29 (12) (2010) 1845–1958.
[23] M. Mori, H. Chen, B. Yao, C. Cheng, A multiple level network approach for clock skew minimization with process variations, in: Proceedings of the IEEE Asia and South Pacific Design Automation Conference, January 2004, pp. 263–268.
[24] H. Su, S.S. Sapatnekar, Hybrid structured clock network construction, in: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, November 2001, pp. 333–336.
[25] M.A.B. Jackson, A. Srinivasan, E.S. Kuh, Clock routing for high-performance ICs, in: Proceedings of the IEEE/ACM Annual Design Automation Conference, June 1990, pp. 573–579.
[26] X. Ye, P. Li, F. Liu, Practical variation-aware interconnect delay and slew analysis for statistical timing verification, in: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, November 2006, pp. 54–59.
[27] D.S. Johnson, Approximation algorithms for combinatorial problems, Journal of Computer and System Sciences 9 (3) (1974) 256–278.

Ameer Abdelhadi is a Ph.D. candidate at the Department of Electrical and Computer Engineering, University of British Columbia, Canada. He received his B.Sc. and M.Sc. in Electrical and Computer Engineering from the Technion—Israel Institute of Technology in 2007 and 2010, respectively. He held design and research positions in the semiconductors industry between 2003 and 2008. He served as a research and teaching assistant in the Technion Electrical Engineering Department between 2008 and 2010, and in the University of British Columbia Electrical and Computer Engineering Department since 2011. His research interests include reconfigurable computing, VLSI design methodologies, computer-aided-design for VLSI circuits and PLDs, clocking techniques and asynchronous circuits.

Ran Ginosar received his BSc from the Technion—Israel Institute of Technology in 1978 (*summa cum laude*) and his Ph.D. from Princeton University, USA, in 1982, both in Electrical and Computer Engineering. He worked at AT&T Bell Laboratories in 1982–1983, and joined the Technion faculty in 1983. He was a visiting Associate Professor with the University of Utah in 1989–1990, and a visiting faculty with Intel Research Labs in 1997–1999. He is an Assoc. Professor with the Departments of Electrical Engineering and Computer Science and serves as Head of the VLSI Systems Research Center at the Technion. His research interests include VLSI architecture, many-core computers, asynchronous logic and synchronization, networks on chip and biologic implant chips. He has co-founded several companies in various areas of VLSI systems.

Avinoam Kolodny received his doctorate in microelectronics from Technion—Israel Institute of Technology in 1980. He joined Intel Corporation, where he was engaged in research and development in the areas of device physics, VLSI circuits, electronic design automation, and organizational development. He has been a member of the Faculty of Electrical Engineering at the Technion since 2000. His current research is focused primarily on interconnects in VLSI systems, at both physical and architectural levels.

Eby G. Friedman received the B.S. degree from Lafayette College in 1979, and the M.S. and Ph.D. degrees from the University of California, Irvine, in 1981 and 1989, respectively, all in electrical engineering.

From 1979 to 1991, he was with Hughes Aircraft Company, rising to the position of manager of the Signal Processing Design and Test Department, responsible for the design and test of high performance digital and analog IC's. He has been with the Department of Electrical and Computer Engineering at the University of Rochester since 1991, where he is a Distinguished Professor, and the Director of the High Performance VLSI/IC Design and Analysis Laboratory. He is also a Visiting Professor at the Technion - Israel Institute of Technology. His current research and teaching interests are in high performance synchronous digital and mixed-signal microelectronic design and analysis with application to high speed portable processors and low power wireless communications.

He is the author of over 400 papers and book chapters, 12 patents, and the author or editor of 15 books in the fields of high speed and low power CMOS design techniques, 3-D design methodologies, high speed interconnect, and the theory and application of synchronous clock and power distribution networks. Dr. Friedman is the Regional Editor of the *Journal of Circuits, Systems and Computers*, a Member of the editorial boards of the *Analog Integrated Circuits and Signal Processing*, *Microelectronics Journal*, *Journal of Low Power Electronics*, *Journal of Low Power Electronics and Applications*, and *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, Chair of the *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* steering committee, and a Member of the technical program committee of a number of conferences. He previously was the Editor-in-Chief of the *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, a Member of the editorial board of the *Proceedings of the IEEE*, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, and *Journal of Signal Processing Systems*, a Member of the Circuits and Systems (CAS) Society Board of Governors, Program and Technical chair of several IEEE conferences, and a recipient of the University of Rochester Graduate Teaching Award and a College of Engineering Teaching Excellence Award. Dr. Friedman is a Senior Fulbright Fellow and an IEEE Fellow.