Timing–Driven Variation–Aware Nonuniform Clock Mesh Synthesis

Ameer Abdelhadi, Ran Ginosar, Avinoam Kolodny, and Eby G. Friedman*

Technion – Israel Institute of Technology

Department of Electrical Engineering

* Also with Dept. of Electrical and Computer Engineering, University of Rochester

This research was supported in part by the Technion ACRC and by the ALPHA research consortium

Non-tree Clock Topologies (1)

+ Immune to process, voltage, and temperature (PVT) variations.

+ Tolerate non-uniform switching

+ Tolerate unbalanced loads

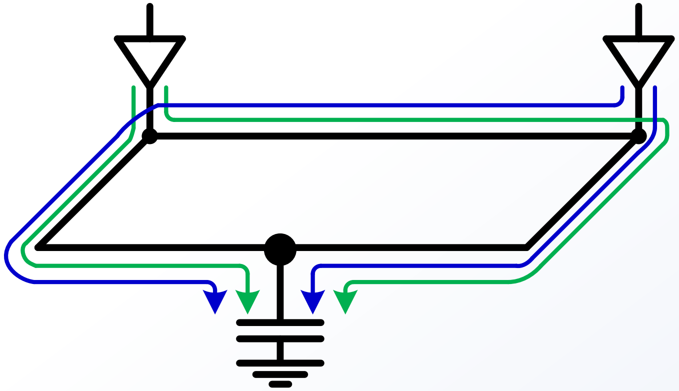
+ Low skew, variations, and jitter

+ Overcome late design changes

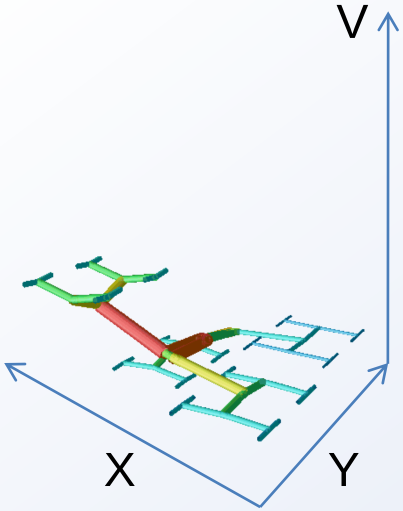
– Difficult to analyze or automate

– Require significant resources

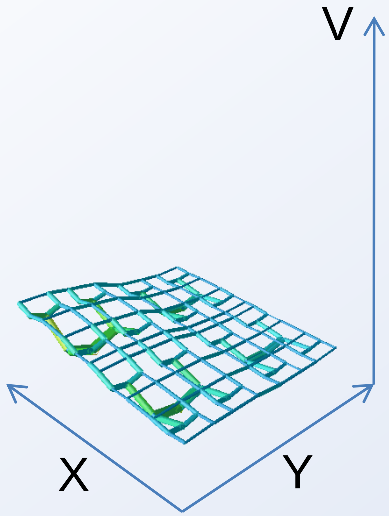
– dissipate higher power, due to:



Multi-path signal propagation



Tree driving 10.6pF load Non-Uniform load at 2GHz

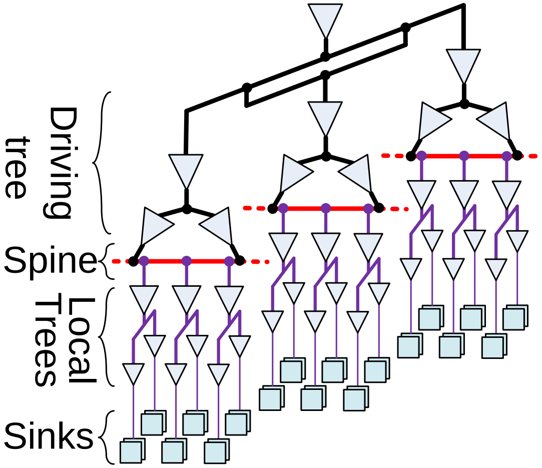


Trees driving a Grid with 68pF Non-Uniform load at 2GHz

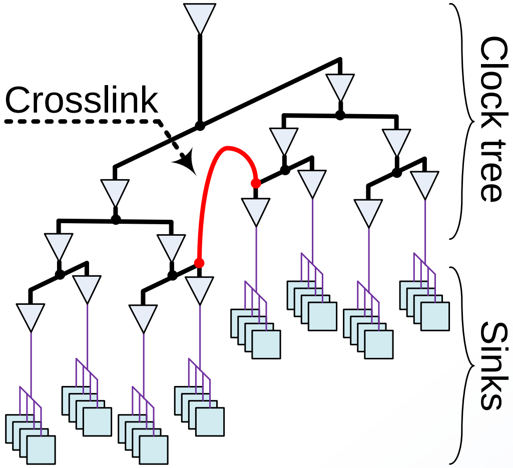
Animations by Phillip J. Restle

P. J. Restle, "Technical visualizations in VLSI design: visualization," Proc. DAC, pp. 494-499, 2001.

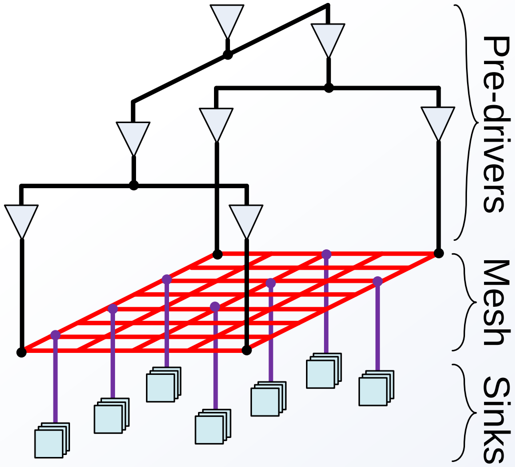
Non-tree Clock Topologies (2)



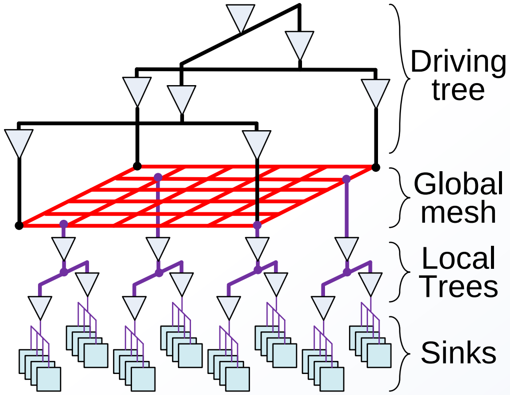
Pentium 4 spine [1]



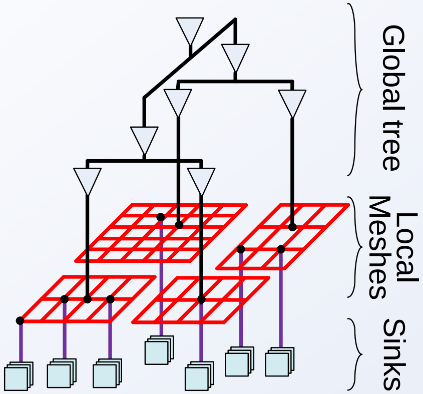
Tree with crosslinks [2]



Leaf level global mesh



Global mesh with local trees (MLT) [3]

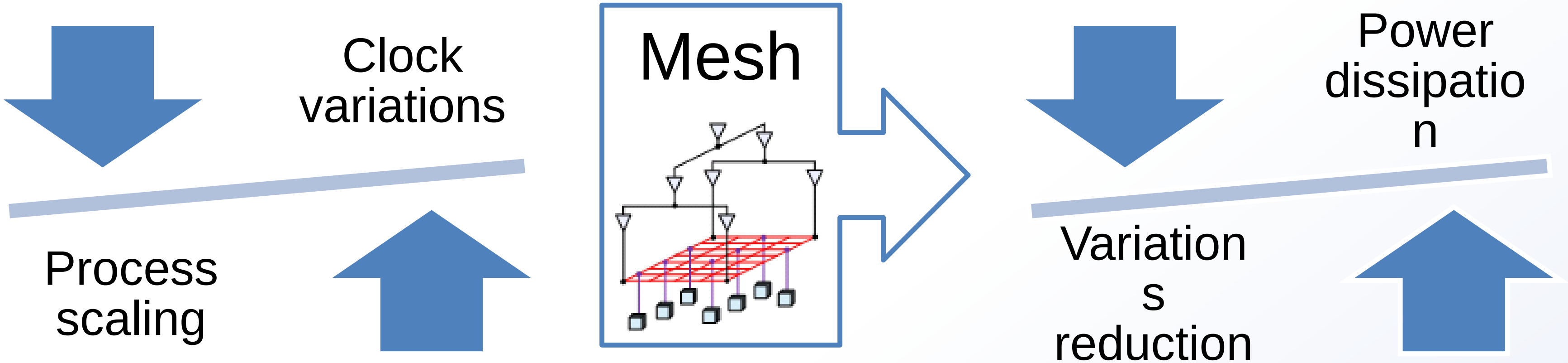


Global tree with local meshes (TLM) [3]

[1] N. A. Kurd et al., "A Multigigahertz Clocking Scheme for the Pentium 4 Microprocessor," JSSC 36(11):1647–1653, 2001.

[2] A. Rajaram, J. Hu, and R. Mahapatra, "Reducing Clock Skew Variability Via Crosslinks," Proc. DAC, pp. 18–

Motivation



Process scaling

Clock variations

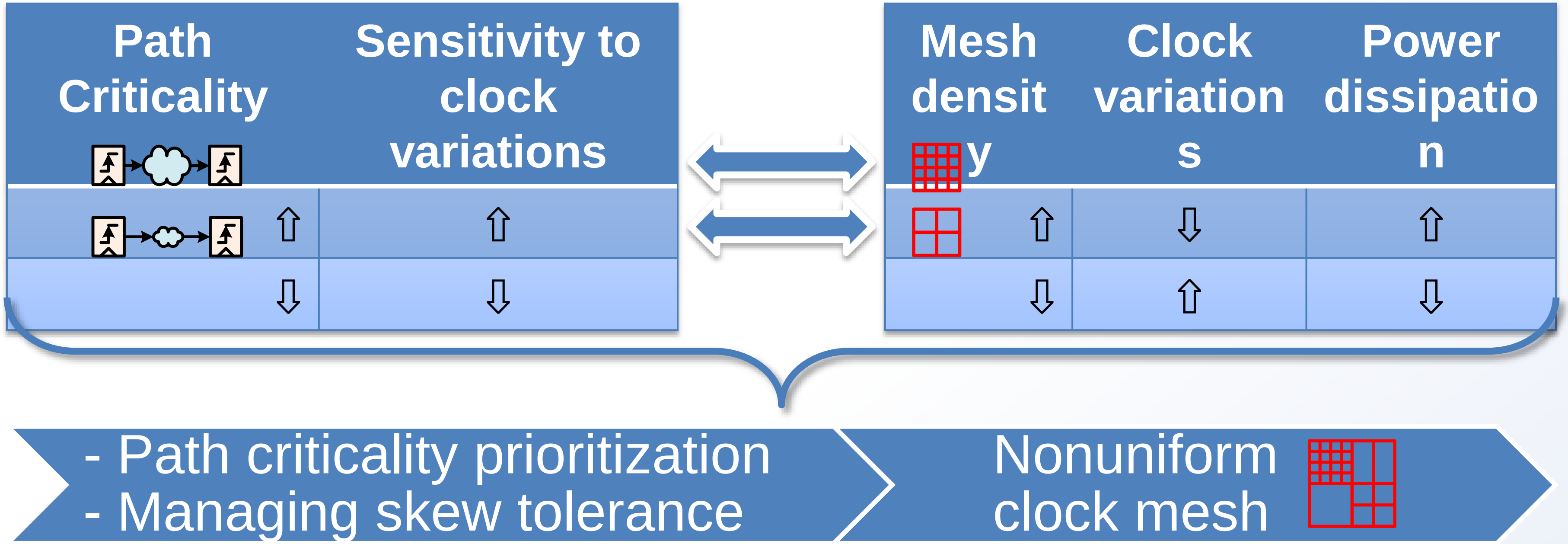
Mesh

Variation
s
reduction

Power
dissipatio
n

<u>Goal:</u> reducing clock skew variations while keeping minimal power dissipation overhead

Method



Path Criticality	Sensitivity to clock variations
⇧	⇧
⇩	⇩

Mesh density	Clock variations	Power dissipation
⇧	⇩	⇧
⇩	⇧	⇩

- Path criticality prioritization
- Managing skew tolerance

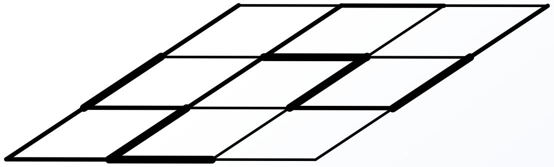
Nonuniform clock mesh

Selective reduction of clock skew variations based on circuit timing criticality

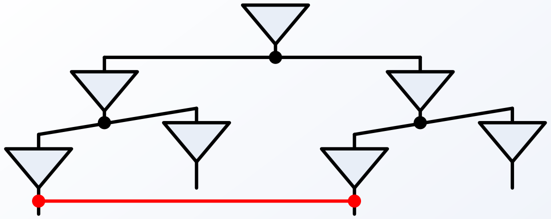
Previous Work

Clock mesh design automation:

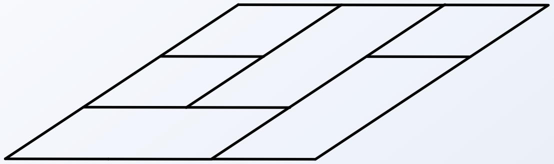
- Segment wire width sizing [1]
- Start from a clock tree and add crosslinks [2],[3]
- Start with a fully uniform mesh, remove redundant segments [4],[5]



Segments sizing



Croslinks



Remove segments

- Circuit timing is not exploited

[1] M. P. Desai, R. Cvijetic, and J. Jensen, "Sizing of Clock Distribution Networks for High Performance CPU Chips," Proc. DAC, pp. 389-394, '96.

[2] A. Rajaram, J. Hu, and R. Mahapatra, "Reducing Clock Skew Variability Via Crosslinks," IEEE TCAD, 25(6): 1176-1182, '06.

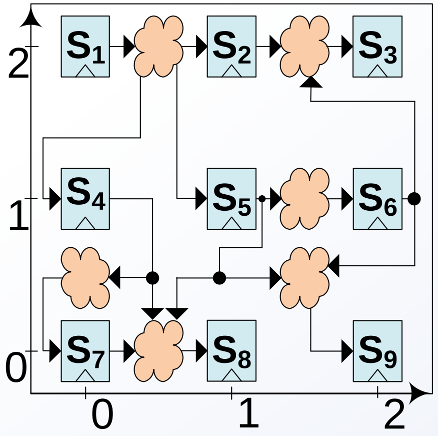
[3] I. Vaisband, R. Ginosar, A. Kolodny, and E. G. Friedman, "Power Efficient Tree-Based Crosslinks for Skew Reduction," Proc. GLSVLSI, pp. 285-290, '09.

[4] A. Rajaram et al., "MeshWorks: an Efficient Framework for Planning, Synthesis and Optimization of Clock Mesh Networks," Proc. ASP-DAC, pp. 250-257, '08.

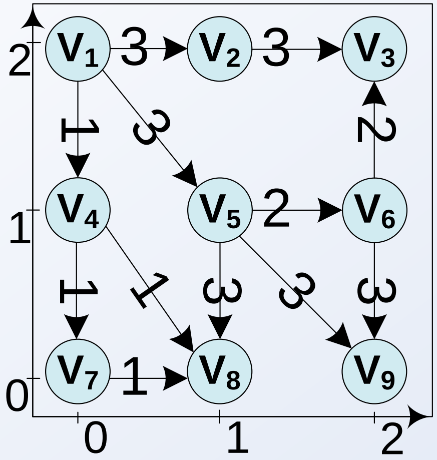
[5] G. Venkataraman, Z. Feng, J. Hu, and P. Li "Combinational Algorithms for Fast Clock Mesh Optimization," Proc. ICCAD, pp. 563-567, '06.

Timing Constraint Graph

- Presents the circuit’s connectivity
- Vertices represent clock sinks:
  $G_C^V=S=\{s_1, s_2, \ldots, s_n\}$
- Edges represent data paths:
  $G_C^E=\{e_{i,j}=v_i\sim v_j \mid P_{delay}^{ij}<\infty, v_i, v_j \in G_C^V\}$
- Edges’ weights are maximum skew constraint (permissible):
  $(\forall e \in G_C^E)\ w_e= skew^{i,j}$
- Vertices also have attributes:
  - Sink capacitance:
    $(\forall v_i \in G_C^V)\ \ C(v_i)=Capacitance(S_i)$
  - Location:
    $(\forall v_i \in G_C^V)\ \ bbox(v_i)=[[x_0,y_0],[x_1,y_1]]$



Floorplan and connectivity



Corresponding graph

Timing–Driven Variation–Aware Problem Formulation

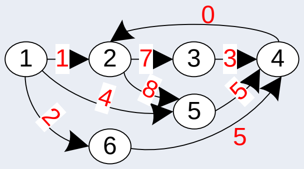
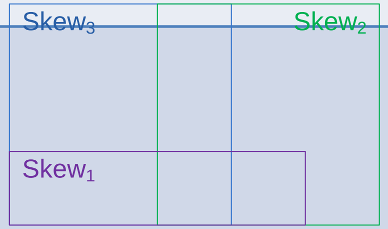
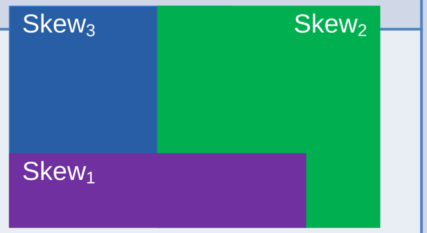
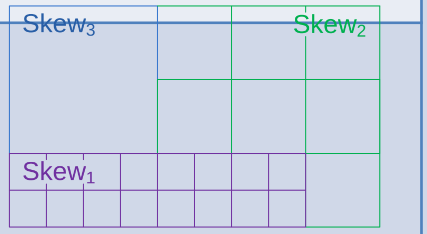
- Given:
  - Circuit connectivity
  - Static timing analysis
- Relative tolerance parameter ξ
  - user defined
  - upper bound of the maximum skew variation ratio overall maximum skew constraints:

$$(\forall e_{i,j} \in G_C^E)\ \xi \geq \delta^{i,j}_{max}/skew^{i,j}$$

While:

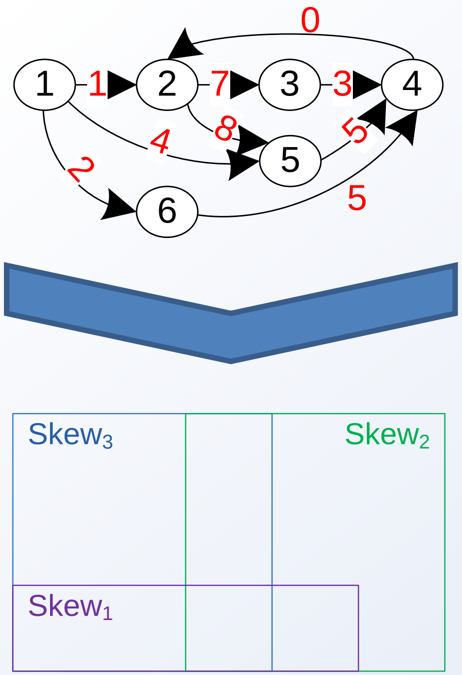
  - $\delta^{i,j}_{max}$ is the maximum skew variation between register Si and Sj
  - $skew^{i,j}$ is the maximum permissible skew between register S_i and S_j
- Construct a clock mesh that satisfies the $\delta^{i,j}_{max}$ requirement

Solution Stages

1	Derive Timing Constraint Graph from Static Timing Analysis (STA)	 1 2 3 4 5 6 1 7 3 0 4 8 5 2 5	TCG
2	Generate skew map ▪ Rectangular shapes ▪ *[skew,cap,bbox]* triples		Floorpaln
3	Remove overlapping ▪ Polygon shapes ▪ *[skew,cap,polygon]* triples		
4	Mesh Generation ▪ Mach a mesh to each polygon ▪ Mesh density satisfies skew variations		

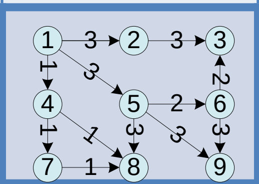
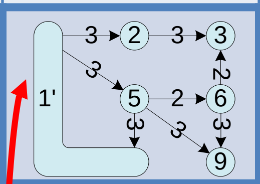
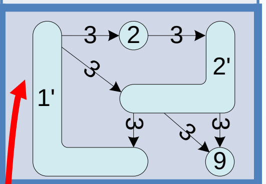
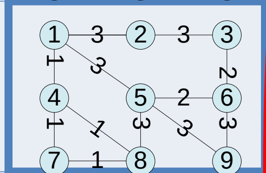
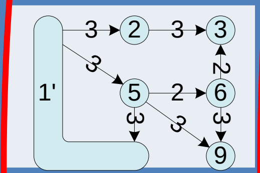
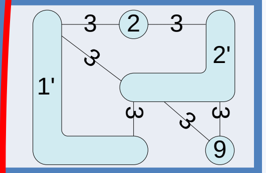
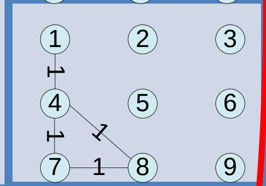
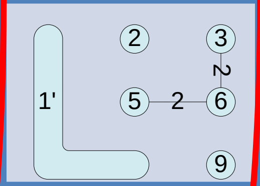
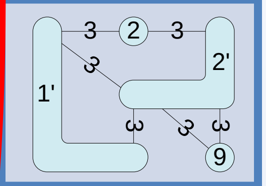
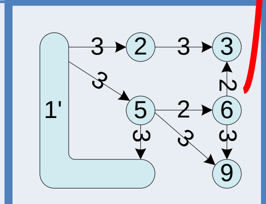
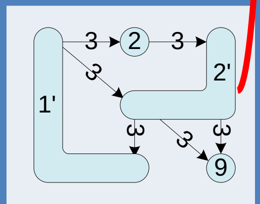
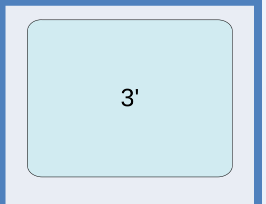
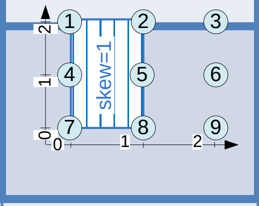
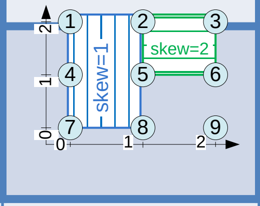
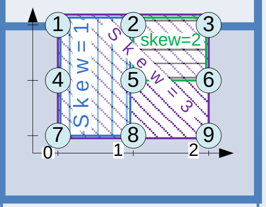
Phase II: Generate Skew Map (1)

- Find regions with deferent skew requirements
- Graph theoretic algorithm
- Inputs:
  - G_c: Constraint Graph
  - T: Thresholds vector
    - Contains skew thresholds
    - Granularity of skew regions
- Output:
  - Skew map with rectangular shapes
  - ***[skew,cap,bbox]*** triples stack
  - Ascending order by skew
- Method:
  - Remove edges with skew lower than threshold
  - Connected components define skew regions
  - Merge connected components with original



Phase II: Generate Skew Map

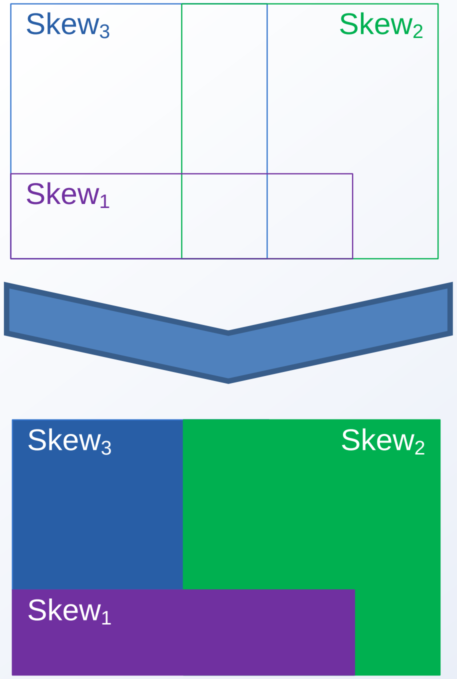
Phase II: Generate Skew Map (2)

***1.** foreach $t \in T$*	*T=[1,2,3]*	**t=1**	**t=2**	**t=3**
***1.1** $G_C^{undirected}$=getUndirected(G_C)*	Initial graph			
	Undirected graph			
***1.2** foreach $e \in G_C^E$* ***1.2.1** if $w_e > t$* *$G_C^{undirected} = G_C^{undirected} / e$*	Remove edges with $w_e > t$			
***1.3** UCC=getConnComp($G_C^{undirected}$)* ***1.4** foreach $ucc \in UCC$* ***1.4.1** v_{merge}=mergeVertices(G_C,ucc)*	Merge connected component			
***1.4.2** $skew_{ucc}$= min(w_e | $e=v_i \sim v_j$;v_i,$v_j \in ucc$)* ***1.4.3** $bbox_{ucc}$=bbox(v_{merge})* ***1.4.4** cap_{ucc}=cap(v_{merge})*	Get *[skew, cap, bbox]* triples			
***1.4.5** push(skewBbox,$skew_{ucc}$,cap_{ucc},$bbox_{ucc}$])*	Push triples into stack	[1,C_1,[[0,0],[1,2]]]	[2,C_2,[[1,1,[2,2]]] [1,C_1,[[0,0],[1,2]]]	[3,C_3,[[0,0],[2,2]]] [2,C_2,[[1,1],[2,2]]] [1,C_1,[[0,0],[1,2]]]

Electronics
Computers
Communications

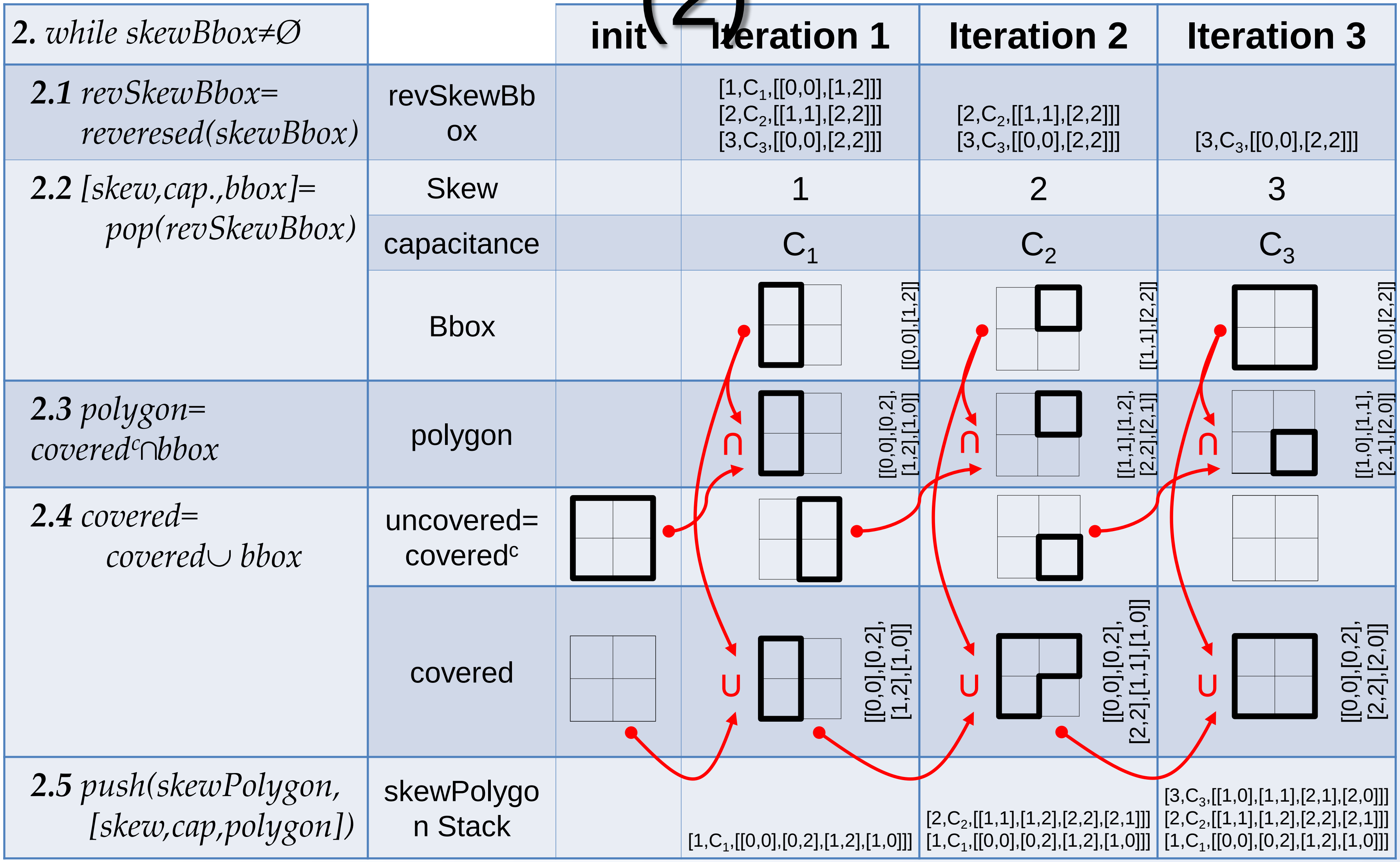
Phase III: Remove Overlapping (1)

- Generate polygons from rectangles
- Geometric algorithm
- Input:
  - *[skew,cap,bbox]* triples stack
    - Generated at phase II
- Output:
  - *[skew,cap,polygon]* triples stack
    - Non-overlapping polygons
- Method:
  - Higher skew regions cover lower regions



Phase III: Remove Overlapping

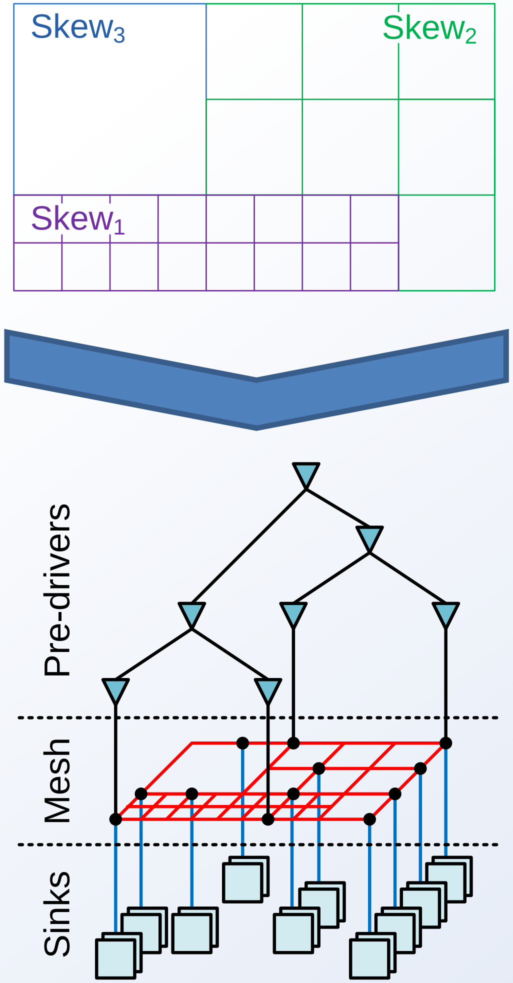
Phase III: Remove Overlapping (2)



2. while skewBbox≠Ø		init	Iteration 1	Iteration 2	Iteration 3
2.1 revSkewBbox= reveresed(skewBbox)	revSkewBbox		[1,C_1,[[0,0],[1,2]]] [2,C_2,[[1,1],[2,2]]] [3,C_3,[[0,0],[2,2]]]	[2,C_2,[[1,1],[2,2]]] [3,C_3,[[0,0],[2,2]]]	[3,C_3,[[0,0],[2,2]]]
2.2 [skew,cap.,bbox]= pop(revSkewBbox)	Skew		1	2	3
	capacitance		C_1	C_2	C_3
	Bbox		[[0,0],[1,2]]	[[1,1],[2,2]]	[[0,0],[2,2]]
2.3 polygon= $covered^c \cap bbox$	polygon		∩ [[0,0],[0,2], [1,2],[1,0]]	∩ [[1,1],[1,2], [2,2],[2,1]]	∩ [[1,0],[1,1], [2,1],[2,0]]
2.4 covered= covered∪ bbox	uncovered= $covered^c$				
	covered		∪ [[0,0],[0,2], [1,2],[1,0]]	∪ [[0,0],[0,2], [2,2],[1,1],[1,0]]	∪ [[0,0],[0,2], [2,2],[2,0]]
2.5 push(skewPolygon, [skew,cap,polygon])	skewPolygon Stack		[1,C_1,[[0,0],[0,2],[1,2],[1,0]]]	[2,C_2,[[1,1],[1,2],[2,2],[2,1]]] [1,C_1,[[0,0],[0,2],[1,2],[1,0]]]	[3,C_3,[[1,0],[1,1],[2,1],[2,0]]] [2,C_2,[[1,1],[1,2],[2,2],[2,1]]] [1,C_1,[[0,0],[0,2],[1,2],[1,0]]]

Phase IV: Mesh Generation

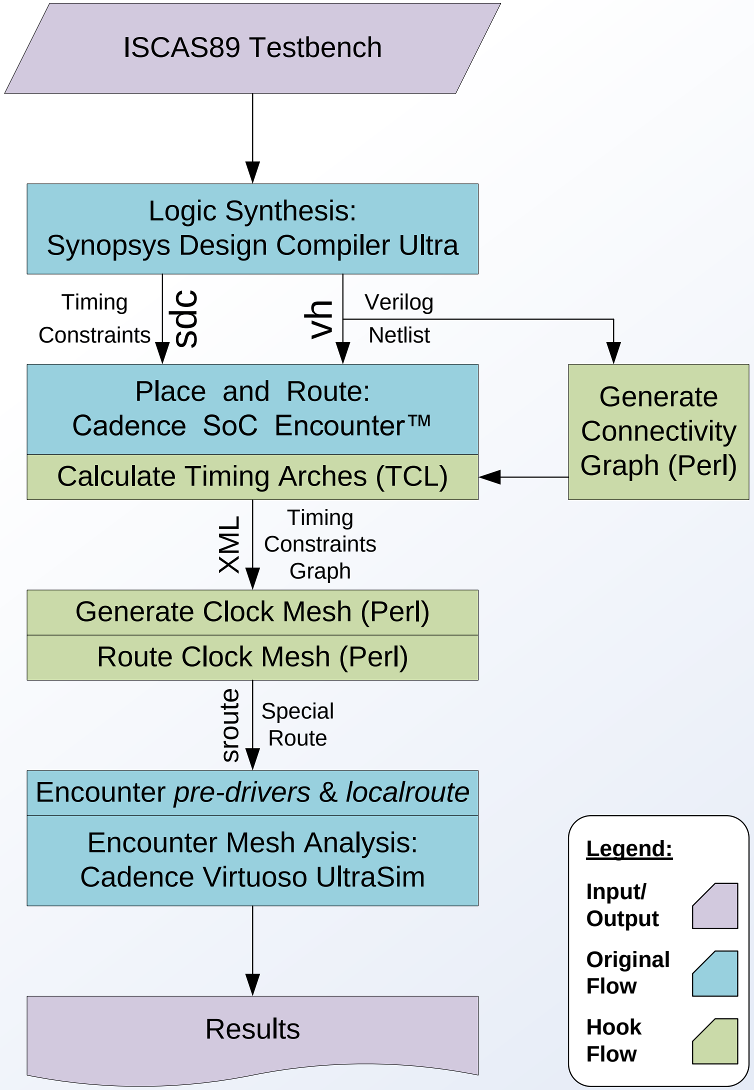
- Mesh to each polygon
- Mesh density is tuned to satisfy skew variations
- Optimized drivers by solving set-covers problem
- Global and local trees are design by the EDA tool clock router



Phase IV: Mesh Generation

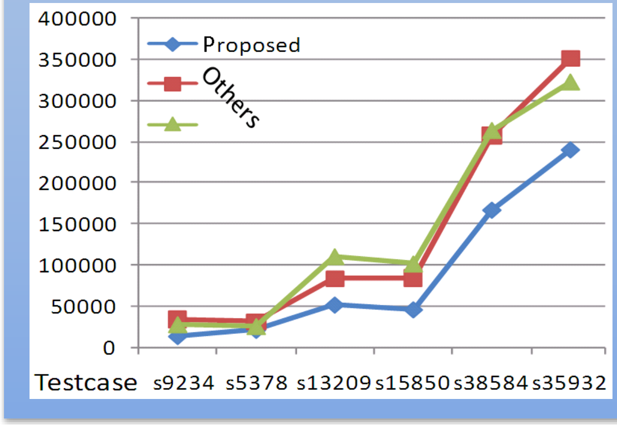
Implementation

- Algorithms:
  - Graph-theoretic
    - for timing constraints
  - Geometric
    - for layout generation
- Quasi-linear (nlogn) runtime
- Design Environment:
  - RTL to layout design flow
  - Standard EDA tools
  - Standard 65nm library
  - ISCAS89 benchmarks

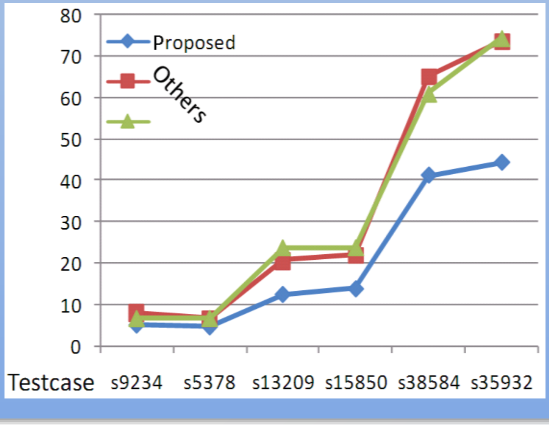

ISCAS89 Testbench
Logic Synthesis: Synopsys Design Compiler Ultra
Timing Constraints
sdc
vh
Verilog Netlist
Place and Route: Cadence SoC Encounter™
Calculate Timing Arches (TCL)
Generate Connectivity Graph (Perl)
XML
Timing Constraints Graph
Generate Clock Mesh (Perl)
Route Clock Mesh (Perl)
sroute
Special Route
Encounter *pre-drivers* & *localroute*
Encounter Mesh Analysis: Cadence Virtuoso UltraSim
Results
Legend:
Input/ Output
Original Flow
Hook Flow

Results

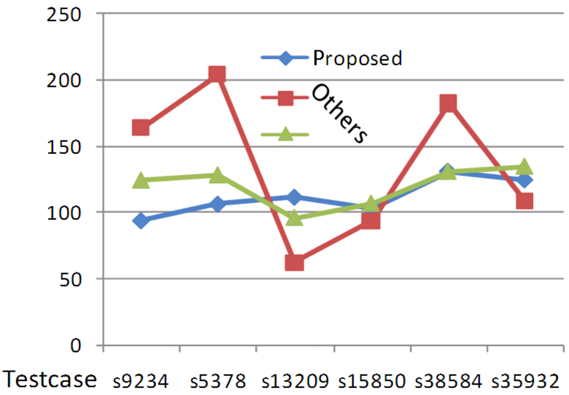
Wire length (um)



Power (mw)



Maximum skew (ps)



37% average reduction in metal

39% average reduction in power dissipation

■ A. Rajaram and D.Z. Pan, "MeshWorks: an efficient framework for planning, synthesis and optimization of clock mesh networks," Proc. ASPDAC, pp. 250-257, 2008.

▲ G. Venkataraman et al. "Combinational algorithms for fast clock mesh optimization," Proc. ICCAD, pp. 563-

Conclusion

Clock mesh design could be effectively automation

Consider non-uniform clock mesh

Consider selective reduction of skew variations based on circuit timing