Optimizing Routability in Large-Scale Mixed-Size Placement

Jason Cong[13], Guojie Luo[23], Kalliopi Tsota[1] and Bingjun Xiao[1]

[1]Computer Science Department, University of California, Los Angeles, USA

[2]School of Electrical Engineering and Computer Science, Peking University, Beijing, China

[3]Joint Research Institute in Science and Engineering by Peking University and UCLA

Routability-Driven Mixed-Size Placement

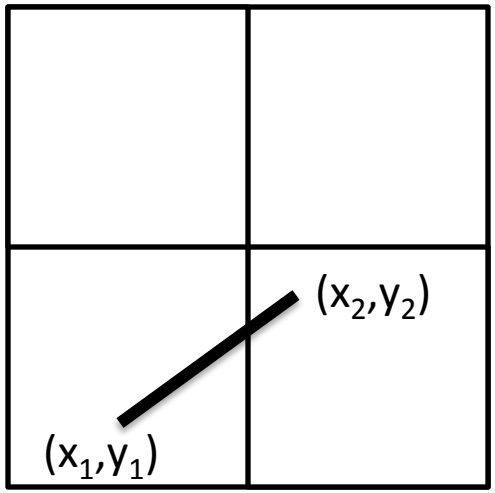
- One of the main objectives is minimization of final routed wirelength
  - Satisfying this objective has a detrimental effect on performance
  - Affects factors such as congestion, delay and timing
- Techniques based on local congestion information have a small impact on quality of final placement
- Targeting congestion during global placement minimizes both routing congestion and final routed wirelength

Proposed Placer

- Multi-level analytical-based routability-driven mixed-size placer
- Alleviation of routing congestion
  - Perform cell inflation to alleviate congested tiles
    - Use cell inflation pattern similar to Ripple [He et al., ICCAD11]
  - Block narrow channels on chip
    - Inflate fixed macros
    - Applied during final level of placement framework
  - Insert dummy cells inside regions of reduced fixed-macro density
    - Applied during final level of placement framework
  - Perform pre-placement inflation at each level of placement framework

Estimation of Routing Congestion

- Divide chip into global tiles and compute congestion of each tile
  - Decompose multi-pin nets into two-pin nets by FLUTE [Chu, ICCAD04]



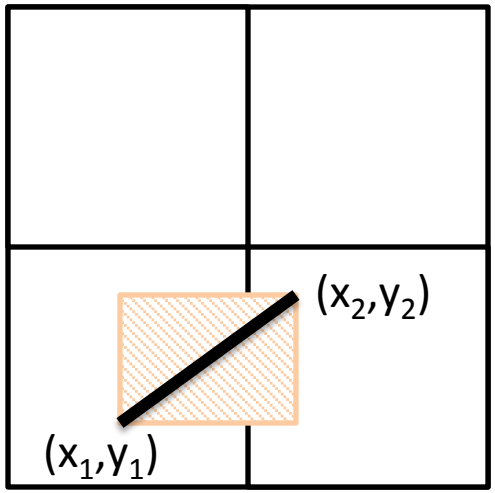
Estimation of Routing Congestion

- Divide chip into global tiles and compute congestion of each tile
  - Decompose multi-pin nets into two-pin nets by FLUTE [Chu, ICCAD04]
  - Horizontal direction

$$SupplyH = (TileWidth)(TileHeight) - BlockageH$$

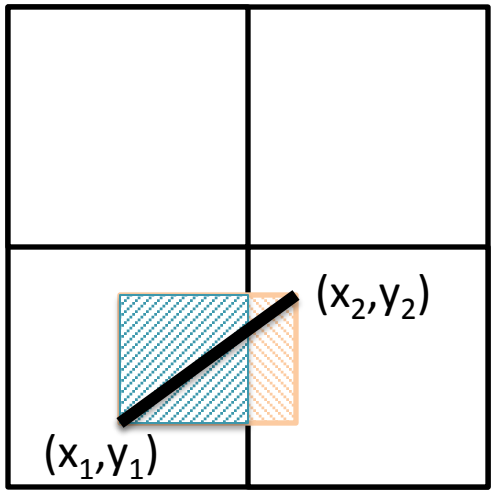
$$DemandH = \frac{(Ovlp)(WireH)}{(WidthBB)(HeightBB)}$$

$$CongestionH = \frac{SupplyH - DemandH}{SupplyH}$$



Estimation of Routing Congestion

- Divide chip into global tiles and compute congestion of each tile
  - Decompose multi-pin nets into two-pin nets by FLUTE [Chu, ICCAD04]
  - Horizontal direction



$$SupplyH = (TileWidth)(TileHeight) - BlockageH$$

$$DemandH = \frac{(Ovlp)(WireH)}{(WidthBB)(HeightBB)}$$

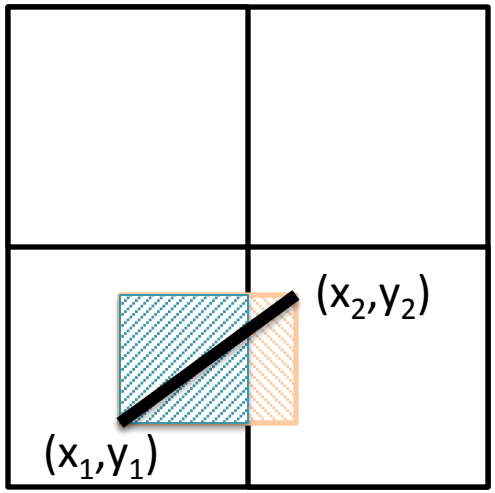
$$CongestionH = \frac{SupplyH - DemandH}{SupplyH}$$

$$WireH = \left(\max\{x_1 - x_2\} - \min\{x_1 - x_2\}\right) WireSpaceH$$

$$WireSpaceH = \frac{TileHeight}{\sum_{i=1}^{LayerNum} HTrack(i)}$$

Estimation of Routing Congestion

- Divide chip into global tiles and compute congestion of each tile
  - Decompose multi-pin nets into two-pin nets by FLUTE [Chu, ICCAD04]
  - Horizontal direction
  - Vertical direction



$$SupplyH = (TileWidth)(TileHeight) - BlockageH$$

$$DemandH = \frac{(Ovlp)(WireH)}{(WidthBB)(HeightBB)}$$

$$CongestionH = \frac{SupplyH - DemandH}{SupplyH}$$

$$SupplyV = (TileWidth)(TileHeight) - BlockageV$$

$$DemandV = \frac{(Ovlp)(WireV)}{(WidthBB)(HeightBB)}$$

$$CongestionV = \frac{SupplyV - DemandV}{SupplyV}$$

Routability Metric

- Metric of DAC 2012 routability-driven placement contest
  - Accounts for both routability and runtime
  - ACE – Average congestion of g-cell edges based on histogram of g-edge congestion [Wei et al., DAC12]
  - ACE(x) – Average congestion of top x% congested g-cell edges
  - PWC – Peak weighted congestion
  - RC – Routing congestion
  - PF – Penalty factor that scales HPWL to account for routing congestion

Routability Metric

- Metric of DAC 2012 routability-driven placement contest
  - Accounts for both routability and runtime
  - ACE – Average congestion of g-cell edges based on histogram of g-edge congestion [Wei et al., DAC12]
  - ACE(x) – Average congestion of top x% congested g-cell edges
  - PWC – Peak weighted congestion
  - RC – Routing congestion
  - PF – Penalty factor that scales HPWL to account for routing congestion

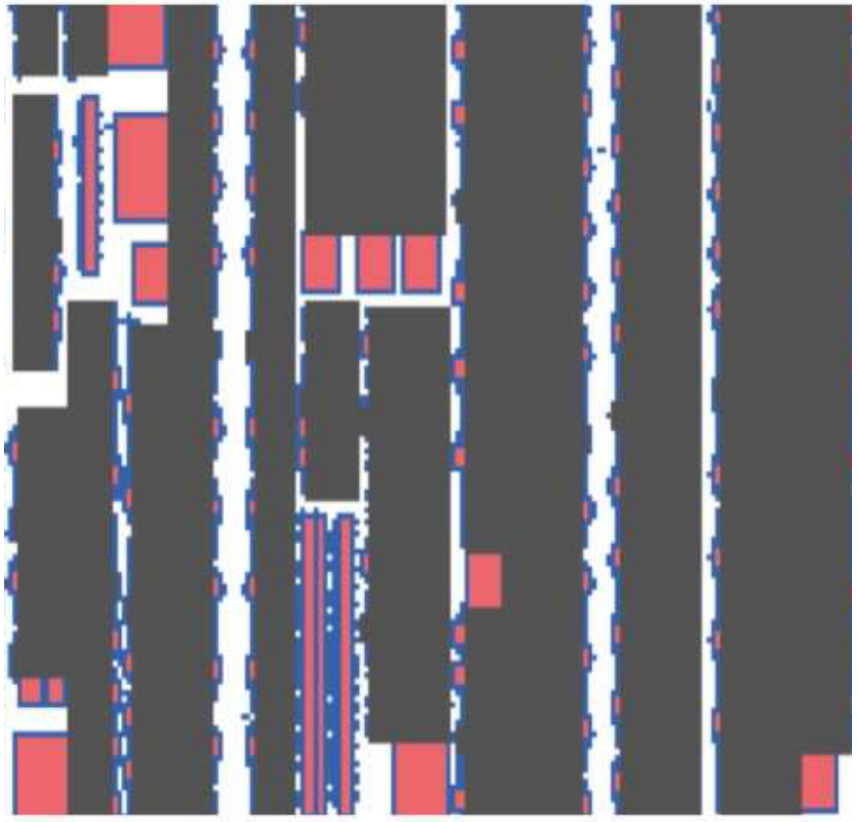
$$PWC = \frac{ACE(x)}{4}, x \in 0.5,1,2,5$$

$$RC = \max(100, PWC)$$

$$ContestMetric = HPWL\left(1 + PF\left(RC - 100\right)\right)\left(1 + RunTimeFactor\right)$$

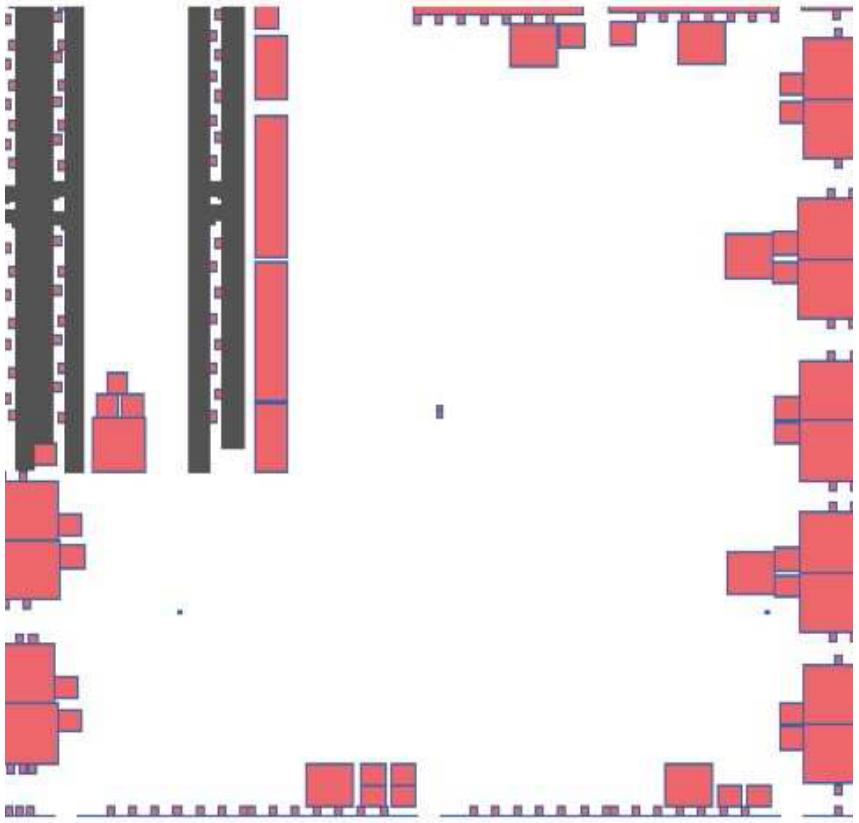
Narrow Channel Reduction by applying Neighbor-Based Fixed-Macro Inflation

- Locations of fixed macros form narrow channels
- Existence of narrow channels contributes to routing congestion
  - Movable cells trapped inside narrow channels
- Reduce routing congestion by blocking narrow channels



Dummy Cell Insertion inside Regions of Reduced Fixed-Macro Density

- Existence of large empty regions on chip contributes to routing congestion
- Insert dummy cells inside large empty regions of the design

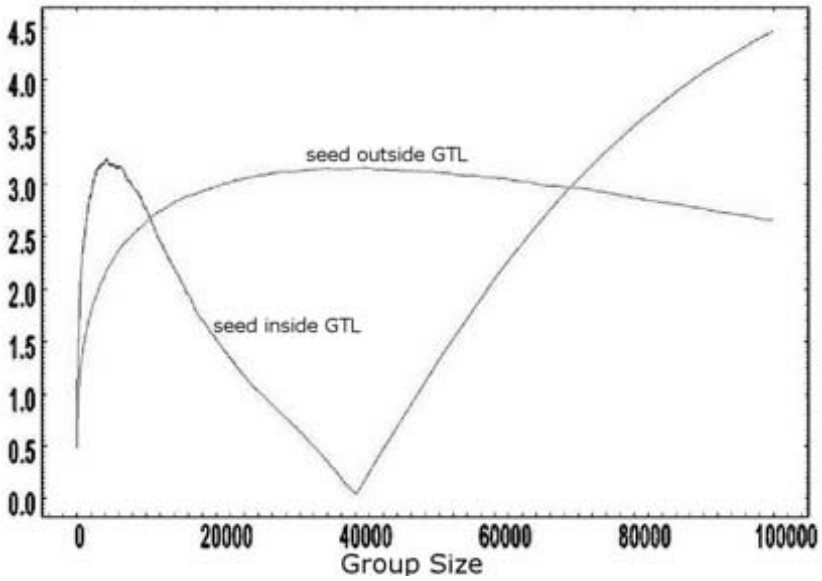


Pre-Placement Inflation
I. GTL-Based Inflation

- Adopt group of tangled logic (GTL) metric
  - [Jindal et al., DAC10]
  - Property of a tangled metric
    - Strong internal connectivity
    - Weak external connectivity
    - Independent of cluster size
  - C: cell cluster
  - T(C): net cut of C (external connectivity)
  - GTL-S(C) = $T(C) / |C|^p$
    - a constant of an average cluster according to Rent's rule
  - nGTL-S(C) = $T(C) / (A(G) |C|^p)$
    - A(G) : average pin count per cell in the circuit G
    - less bias by the pin count in the library
  - GTL-SD(C) = $T(C) / (A(G) |C|^{p\, A(C) / A(G)})$
    - A(C): average pin count per cell in the cluster C
    - emphasize the internal connectivity

Pre-Placement Inflation
I. GTL-Based Inflation (cont.)

- Curve contains distinct trough if tangled logic appears during the growth
- Position of trough indicates when cluster growth has reached the most tangled logic structure
- GTL-based technique for cluster growth imposes additional runtime overhead



Pre-Placement Inflation
I. GTL-Based Inflation (cont.)

- [Jindal et al., DAC10]
  - Start from a seed as the initial cluster
  - Select a strongest connected cell to add to the cluster
  - Evaluate the GTL metric until reaching the trough
- Disadvantage
  - Finding a strongest connected cell is slow
  - Need to repeat multiple times to find more tangled structure
- Our implementation
  - Does not apply a stand-alone tangled structure detection step
  - Performs GTL metric evaluation during the multi-level clustering
  - Each cluster maintains a curve of GTL metric
- Advantage
  - Many tangled structures obtained in one round of clustering
  - Only evaluate the GTL metric for the upcoming clustered cells

Pre-Placement Inflation
I. GTL-Based Inflation (cont.)

- To solve increasing runtimes
  - Incorporate cluster growth method into multi-level placement engine
  - Integrate GTL scoring into clustering process and associate each object with a GTL score curve
- Allocation of whitespace among detected tangled logic structures
  - If GTL score curve of C has large trough width, C is a more tangled logic structure
  - Clusters with smaller area and a large number of cells more likely to be congested and should be inflated

$$Weight(C) = TroughWidth(C)\frac{|C|^2}{Area(C)}$$

Pre-Placement Inflation
II. Pin Density-Based Inflation

- Problem
  - Given a fixed amount of white space as the inflation budget
  - Find an inflation to minimize the maximum pin density per cell
- Algorithm
  - Select the cells with the currently largest pin density
  - Inflate them to match the previously second largest pin density
  - Repeat until the white space budget is used up
  - (Sketch of Proof)

    Any solution with smaller maximum pin density consumes more white spaces than the budget

$$\min \max_i \left\{ \frac{p_i}{A_i^{'}} \right\} \text{ s.t.}$$

$$\mathrm{A}_i \le A_i^{'}, \forall i$$

$$\sum_i \left(A_i^{'} - A_i \right) = W$$

Experimental Results I

	Contest Ripple	Contest NTUplace4	Contest mPL12	Contest simPLR	Our Placer
Circuit	Scaled WL (xE8)	Scaled WL (xE8)	Scaled WL (xE8)	Scaled WL (xE8)	Scaled WL (xE8)
sb19	1.70	1.53	2.46	1.66	1.51
sb14	2.31	2.26	2.67	2.48	2.45
sb16	2.74	2.80	3.01	3.47	2.74
sb9	2.97	2.55	3.22	2.75	2.50
sb3	4.27	3.62	4.66	3.90	3.60
sb11	3.58	3.42	4.52	3.98	3.40
sb6	3.56	3.42	3.95	3.53	3.40
sb2	7.39	6.24	1.33	8.24	6.14
sb12	3.42	3.12	5.40	3.63	3.04
sb7	4.45	3.99	5.24	1.73	3.95
avg.	**1.09**	**1.00**	**1.41**	**1.46**	**0.99**

Experimental Results II

	Contest Ripple	Contest NTUplace4	Contest mPL12	Contest simPLR	Our Placer
Circuit	RunTime (s)	RunTime (s)	RunTime (s)	RunTime (s)	RunTime (s)
sb19	2309	8450	11087	981	9911
sb14	2806	9341	10006	1247	7539
sb16	2737	8573	13670	1128	9435
sb9	4307	13129	14910	1821	12736
sb3	5432	14144	19294	2283	12924
sb11	3745	15263	18284	2342	14723
sb6	4944	11179	20508	2484	17121
sb2	6686	17466	23900	3125	18741
sb12	7635	34831	26107	3459	19245
sb7	11285	25983	22233	3025	17243
avg.	**2.37**	**7.24**	**8.67**	**1.00**	**8.07**

Conclusion

- Proposed placer incorporates narrow channel reduction, dummy-cell insertion inside regions of reduced fixed-macro density and pre-placement inflation
- Reduces routing congestion and improves routability of large-scale mixed-size designs
- Placement tool evaluated using global routers of the DAC 2012 placement contest
- Results compare favorably to the top four teams that participated in the contest