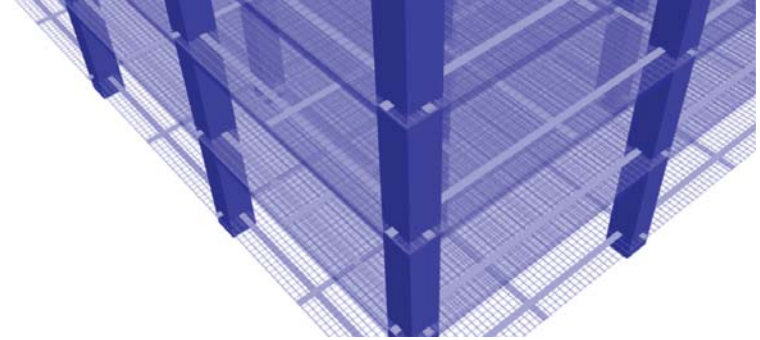
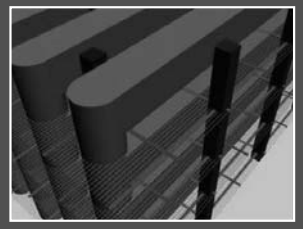
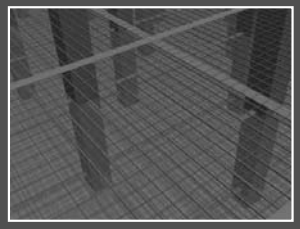


The Law of Attraction: Affinity-Aware Placement Optimization using Graph Neural Networks

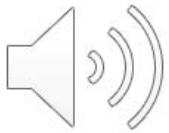

Yi-Chen Lu, Sai Pentapati and Sung Kyu Lim

Georgia Institute of Technology Computer-Aided Design Lab (GTCAD)

*sponsored by Center for Advanced Machine Learning in Electronics (CAEML)

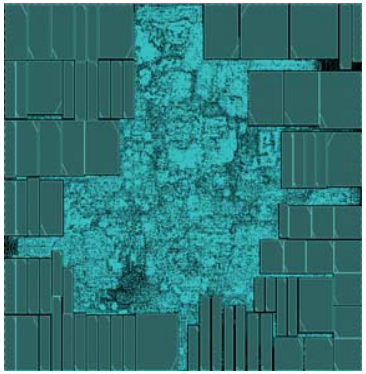
Outline

- Motivations
  - Problem, Objective, Challenge
- Placement Guidance
  - A modern placement optimization technique
- Framework Overview
  - GNN Learning, K-Means Clustering, Placement Opt.
- Graph Neural Networks (GNNs)
  - Initial node features, Message passing scheme
  - Objective function design
- Clustering
  - K-Means clustering
  - Modularity-based clustering
- Experimental Results and Discussion

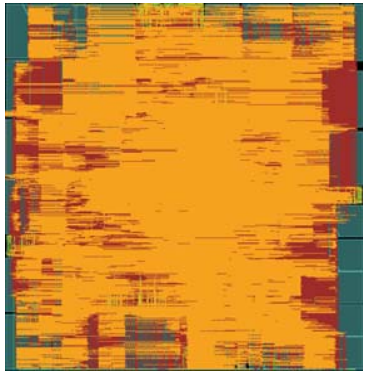


Motivations

- The Placement Problem:
  - Inputs:
    - Constrained Layout L, Design Instances C
  - Goal:
    - Compute a **placement** of instances C onto layout L while optimizing given objectives.
- Objectives: Our Goal
  - Wirelength minimization
  - Timing optimization
  - Routability
  - Low power consumption



Placement

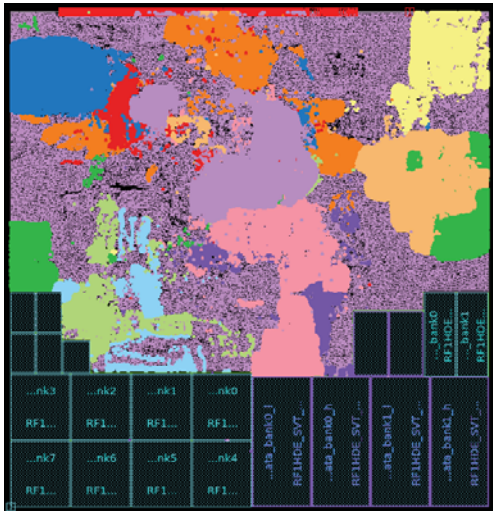


Routing

- Challenges:
  - Traditional placement heuristics are not sufficient for modern huge industrial designs
  - Designers spend significant amount of time in placement iterations

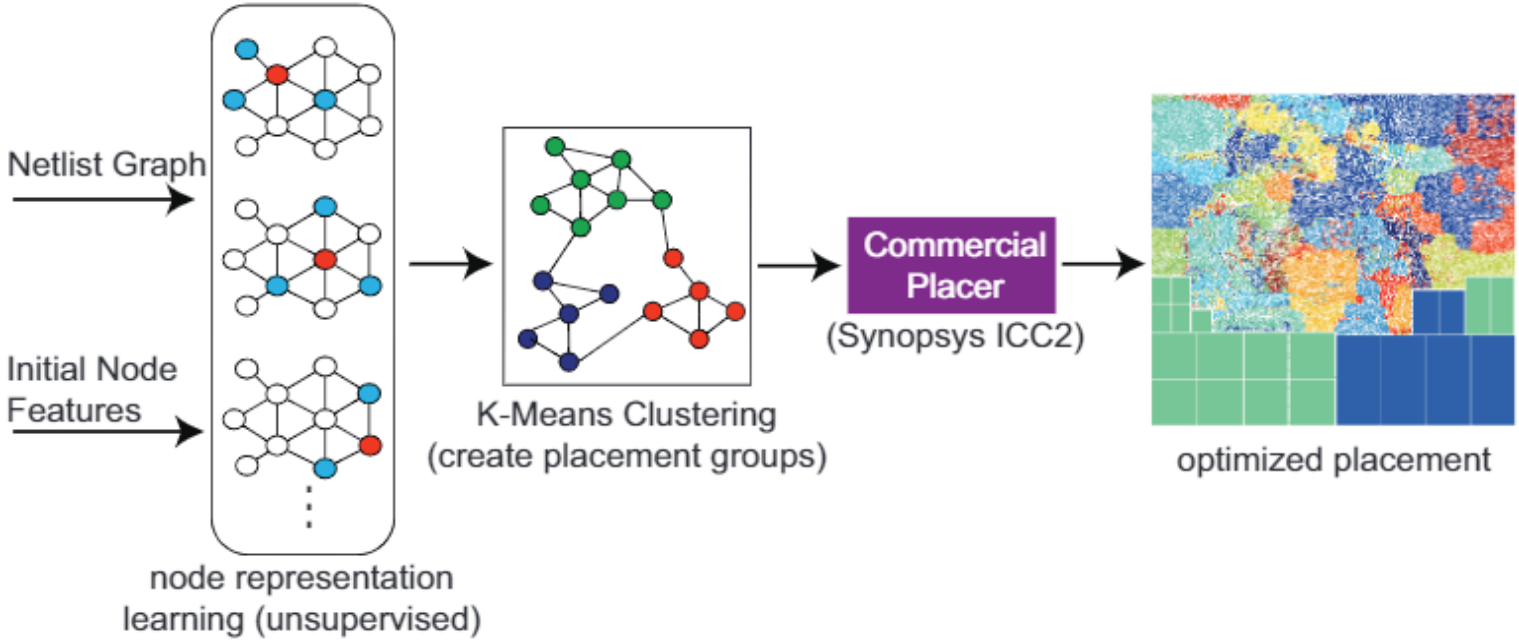
Placement Guidance

- Modern placements are dominated by interconnects
  - Logical affinity among design instances matter a lot
- Key Idea of "Placement Guidance"
  - Group instances into clusters
  - Advise commercial placers to place cells within a cluster together
- How to find good clusters?
  - Design Knowledge
    - Heavily depend on experience → not generalizable
  - Graph Neural Networks (this work)
    - In an unsupervised manner



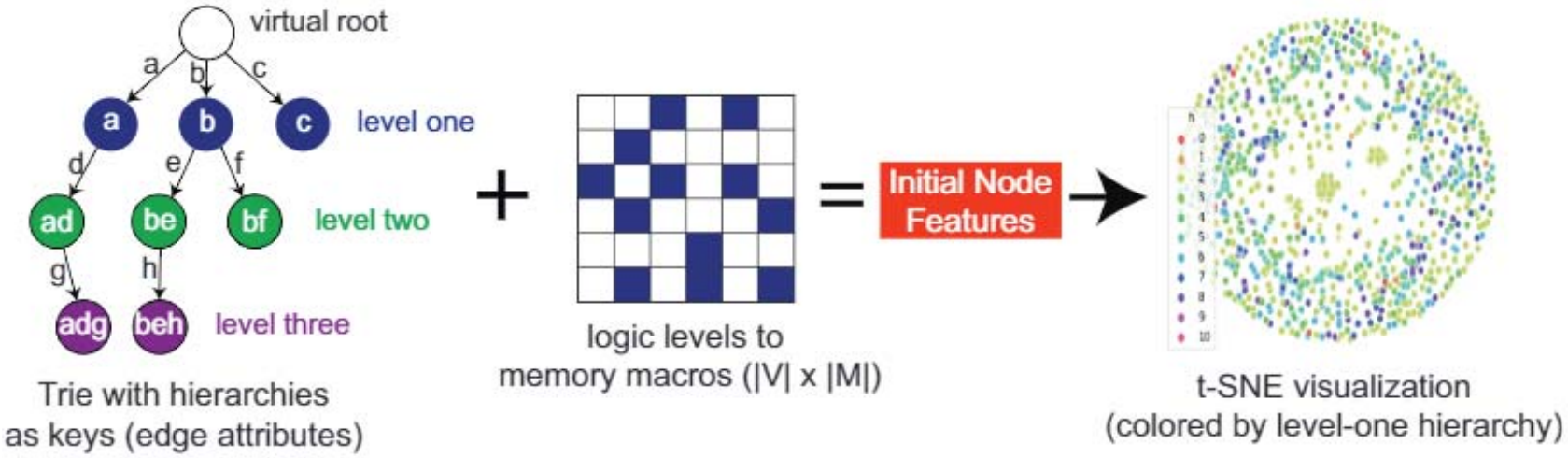
our design
(~200k instances)

Overview of Our Framework



- Stage 1: Graph Learning
  - Node representation learning with unsupervised graphsage
- Stage 2: Similarity-based Clustering
  - K-Means clustering based on learned representations
- Stage 3: Placement Optimization
  - Take grouping information as soft placement constraint in ICC2 placement

Initial Node Features

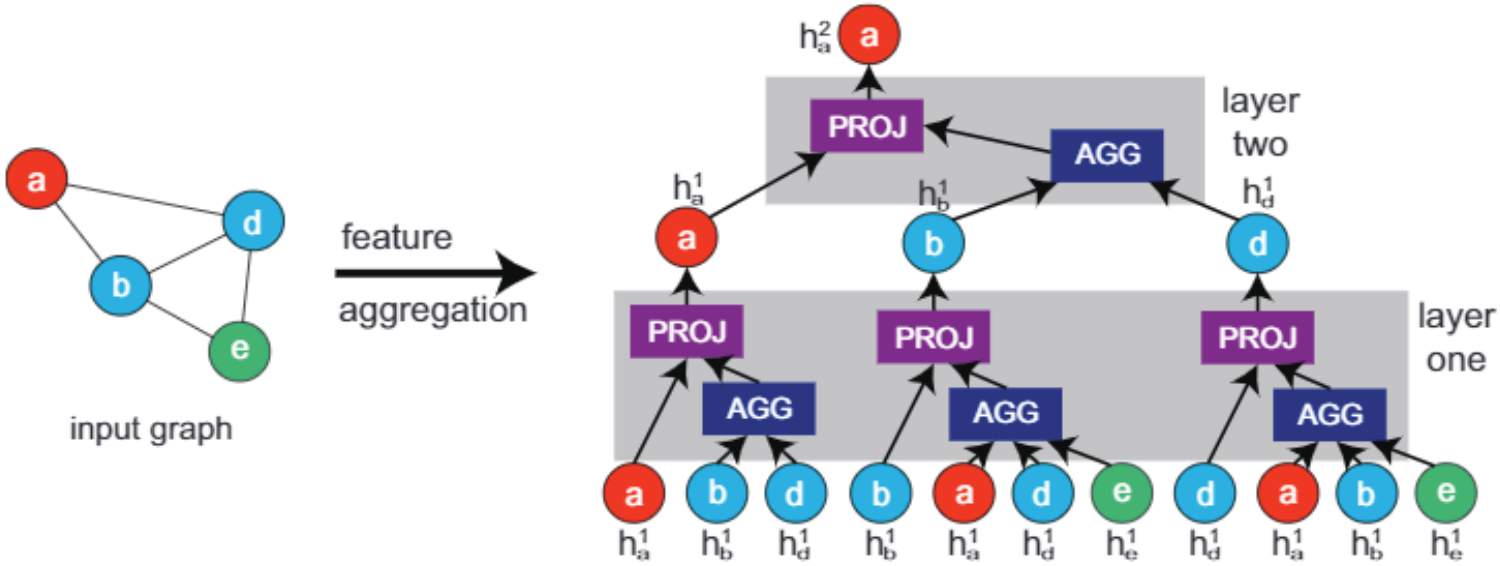


- Hierarchy Information: (in trie structure)
  - A CPU design has several hierarchy levels
  - An instance is initialized as "*hier-a / hier-b / hier-c / inst_name*"
  - Instances within a common hierarchy tend to have more connections
- Memory Macros Affinity:
  - Logic-to-memory paths are often the critical paths
  - Balancing instances with such information is essential

Graph Learning

- GNN can be thought as a set of neural layers
  - In this work, our GNN has two layers with weights $W = \{W_1, W_2\}$

- A node $\boldsymbol{v}$'s feature at level $\boldsymbol{k}$ is obtained by
  - Aggregation: $h_{N(v)}^{k-1} = reduce\_mean\left(W_k^{agg} h_u^{k-1}, \forall u \in N(v)\right)$
  - Projection: $h_v^k = sigmoid(W_k^{proj} concat[h_v^{k-1}, h_{N(v)}^{k-1}])$
  - ($N(v)$ is the neighboring nodes of v)



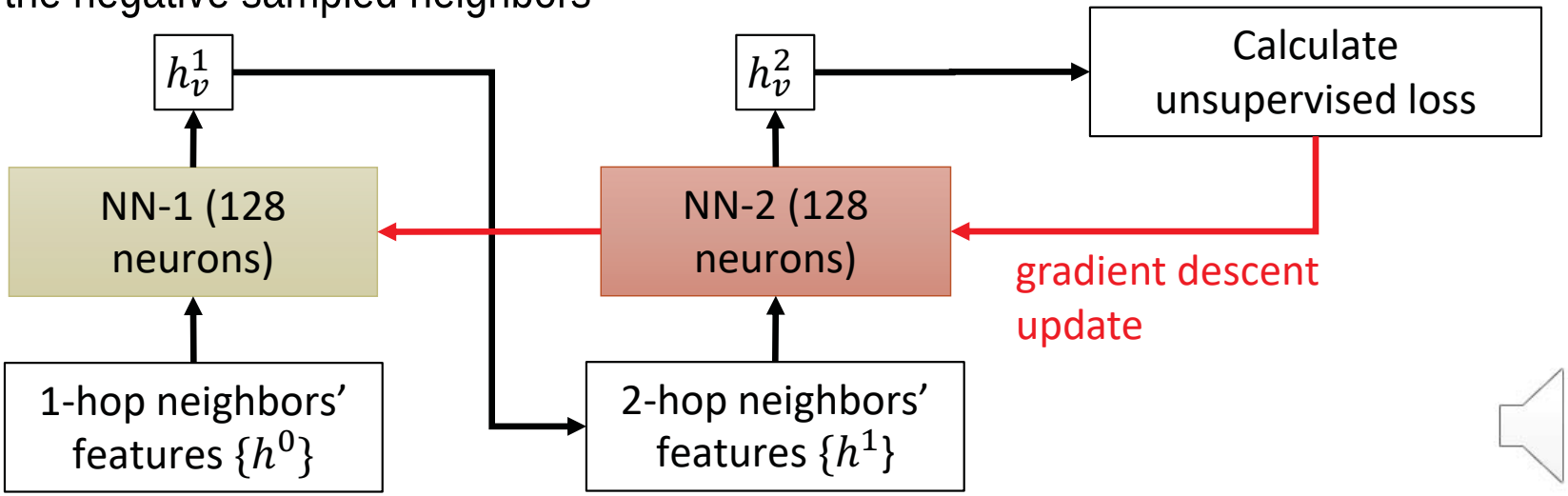
GNN Training / Loss Function Design

- Key Idea:
  - Instances that are logically connected together should better be placed together
    - Should have similar representations
    - Reasonably minimize the wirelength
- Unsupervised loss (node-level):

$$L(y_v) = -\Sigma_{\{u \in N(v)\}} \log(\sigma(y_v^T y_u) - \Sigma_{\{i \in Neg(v)\}} \log(\sigma(-y_v^T y_i)$$

neighboring cells distant cells

where y denotes the learned representations, $N(v)$ denotes the neighbors of v, and $Neg(v)$ denotes the negative sampled neighbors



Our Clustering Approach

- Area-Weighted K-Means clustering
  - Input: learned distributions from GNN
  - Output: standard cell clusters

- How to find “optimal” number of clusters ?
  - Sweep around K to minimize the “Silhouette score”
  - Silhouette score s(v) (node-level):
    - a(v): average distance in a common cluster

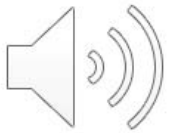
$$a(v) = \frac{1}{|C_v| - 1} \sum_{i \in C_v} ||y_v - y_i||^2$$

    - b(v): min distance to other nodes in different clusters

$$b(v) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{i \in C_k} ||y_v - y_i||^2$$

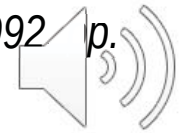
    - s(v): final score

$$s(v) = \frac{b(v) - a(v)}{\max\big(a(v), b(b)\big)}$$



Previous Works

- Placement Prediction:
  - "Finding placement-relevant clusters with fast modularity-based clustering", *M. Fogaça, A. B. Kahng, R. Reis, and L. Wan, ASP-DAC'19*
    - Adopted Louvain [1] to perform fast modularity-based clustering
    - We adopt [1] to perform placement guidance as well
    - [1] "Fast Unfolding of Communities in Large Networks", V. D. Blondel, J. L. Guillaume, R. Lambiotte and E. Lefebvre, J. of Statistical Mechanics: Theory and Experiment 10 (2008)
- Clustering-based Placement Approaches:
  - "Min-cut Floorplacement", *J. A. Roy, S. N. Adya, D. A. Papa and I. L. Markov , IEEE Trans. CAD 25(7) (2006), pp. 1313–1326.*
  - "Performance-Driven Multi-Level Clustering for Combinational Circuits", *C. N. Sze and T.-C. Wang, Proc. ASP-DAC, 2003, pp. 729–734*
  - "Performance-Driven Multi-Level Clustering for Combinational Circuits", *C. N. Sze and T.-C. Wang, Proc. ASP-DAC, 2003, pp. 729–734.*
  - "A Unified Approach to Partitioning and Placement", *R.-S. Tsay and E. S. Kuh, IEEE Trans. Circuits and Systems 38 (1991), pp. 521–533.*
  - "A New Approach to Effective Circuit Clustering", *L. Hagen and A. B. Kahng, Proc. ICCAD, 1992, p. 422–427.*



Experimental Results

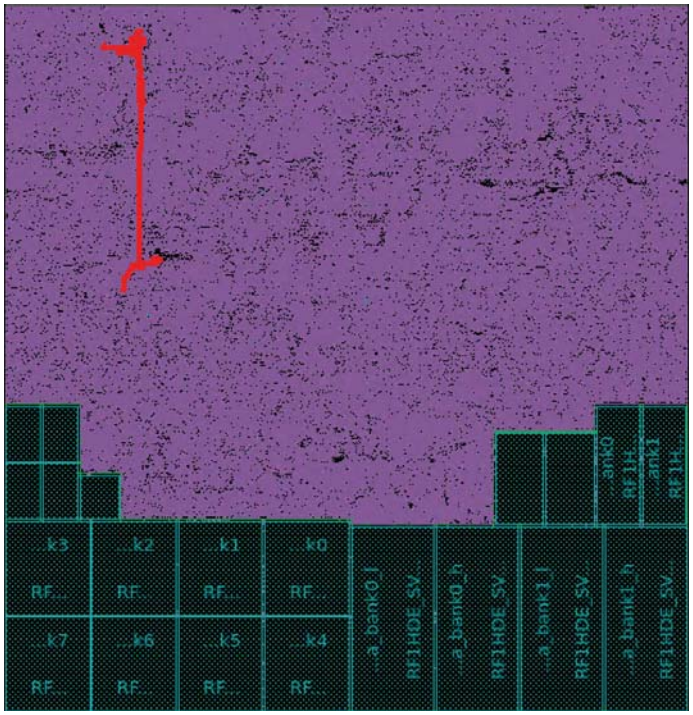
- We validate our results on two CPU core designs:
  - Memory macros are placed (floorplanning) based on design manuals
  - Design Attributes (in TSMC 28nm):

Design Name	# Cells	# Flip-Flops	# Nets	# Macros
CPU-Design-A	202791	22366	206224	21
CPU-Design-B	537085	47552	542391	29

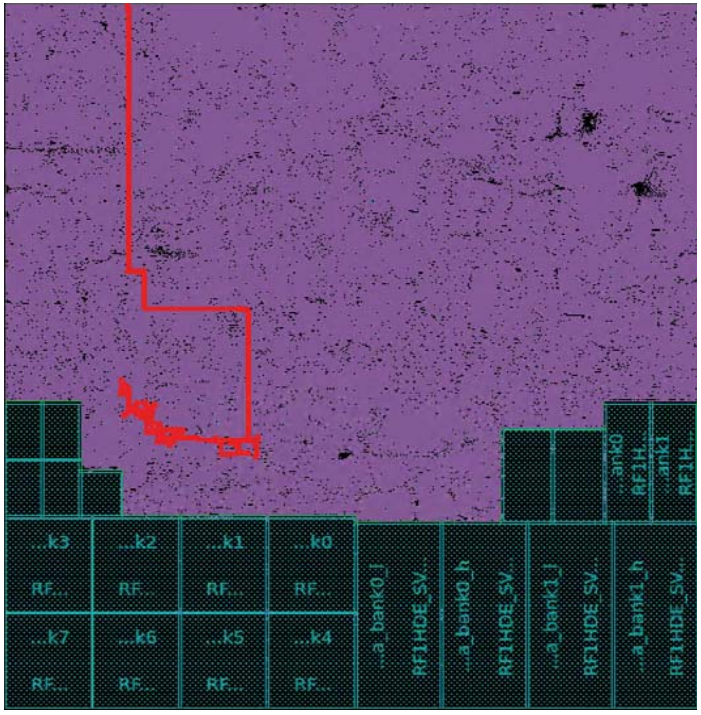
- Final placement results (global + detailed placements)

Design Name	Method	# of clusters	Wirelength (m)	WNS (ns)	TNS (ns)	Total Power (mW)	# inserted buffers
CPU-Design-A	ICC2 default	-	4.37	-0.07	-0.22	142	5942
	Louvain [1]	82	4.34	-0.10	-0.62	141	5826
	ours	22	4.20	-0.01	-0.03	138	5371
CPU-Design-B	ICC2 default	-	11.66	-0.24	-240.39	582	2728
	Louvain [1]	58	11.65	-0.38	-296.54	578	2689
	ours	32	11.55	-0.18	-62.21	574	2274

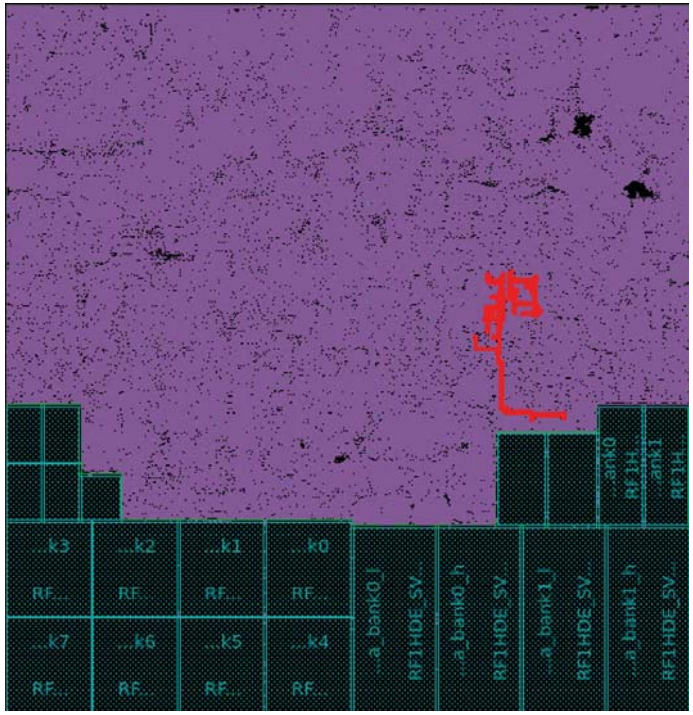
Critical Path Comparison



ICC2 default



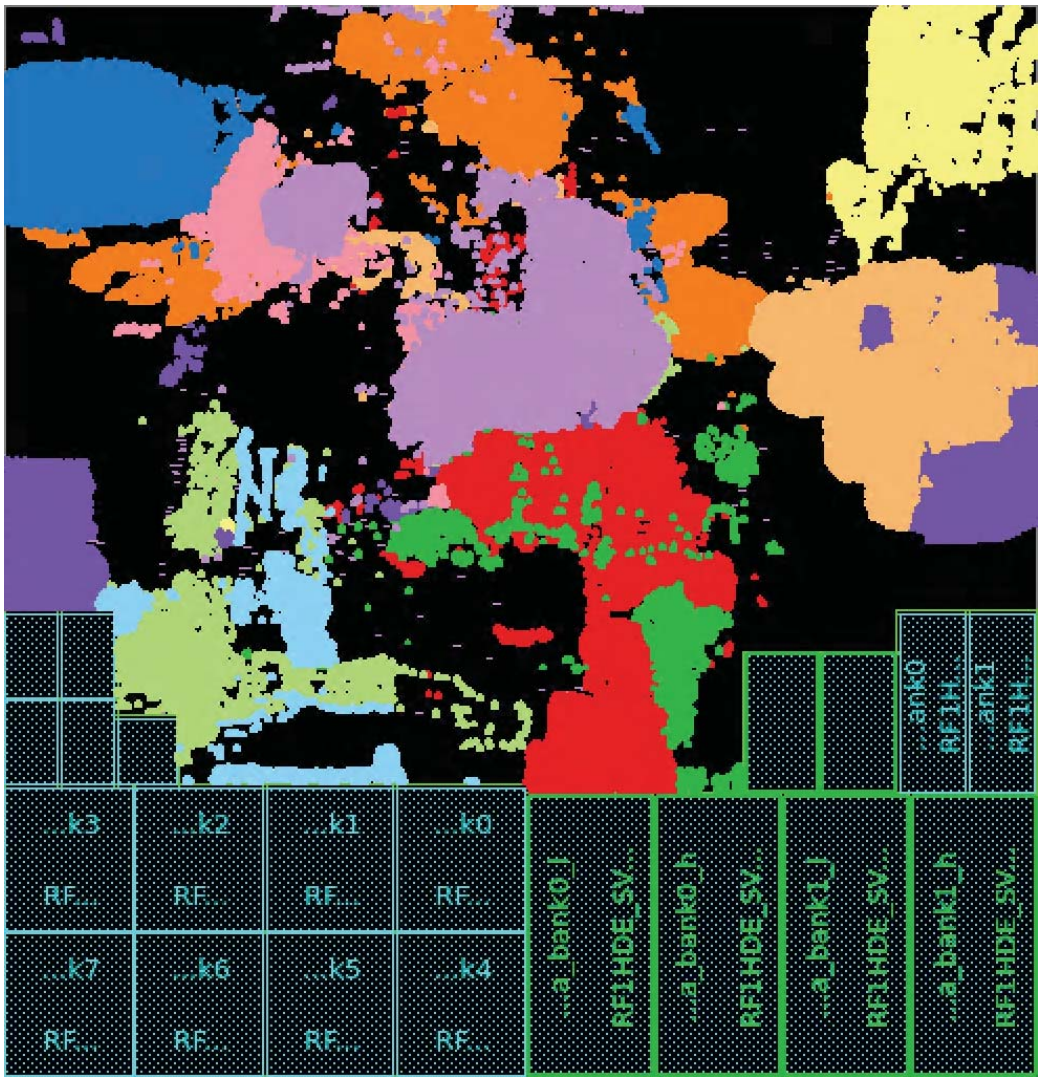
modularity-based



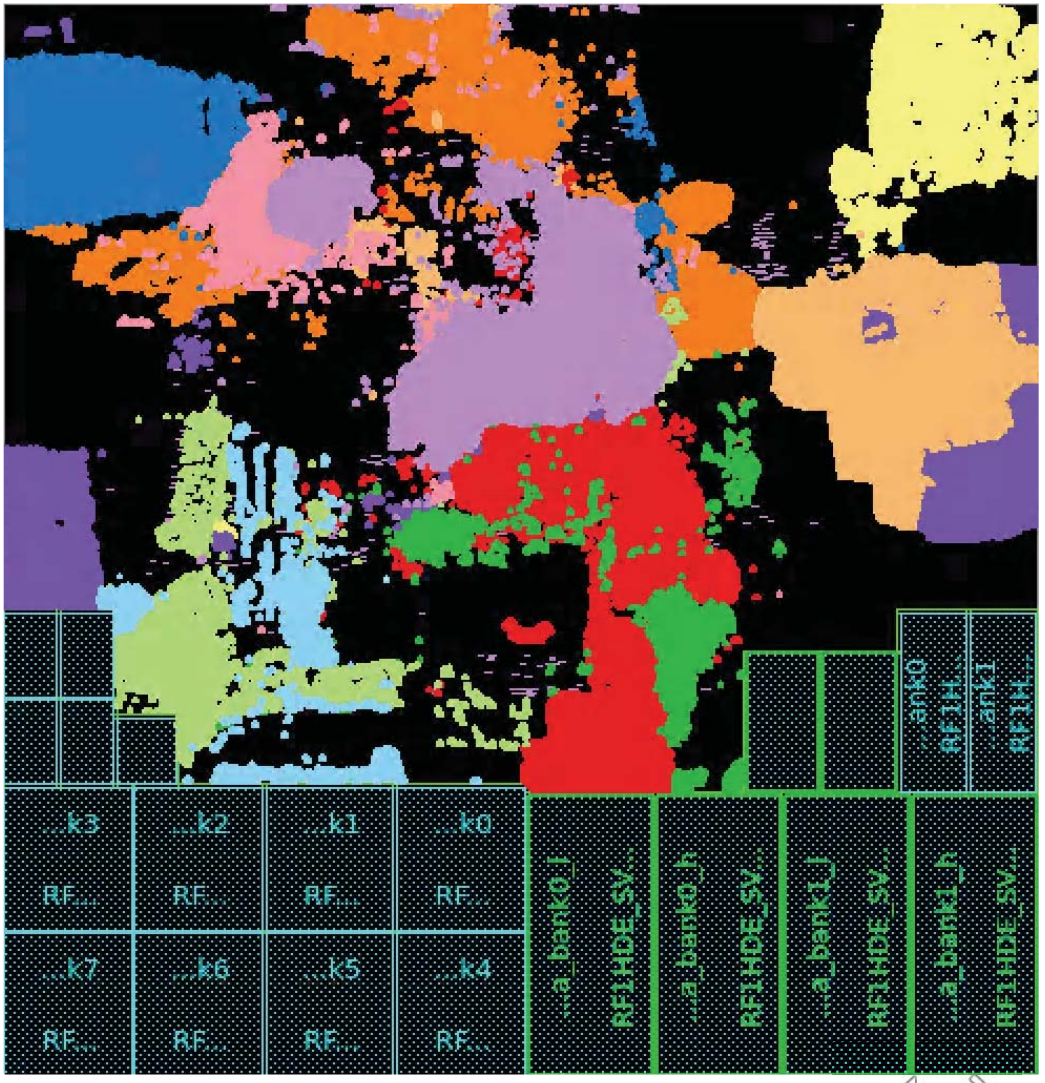
our approach

Design A	ICC2 default	Modularity-based	ours
WNS (ns)	-0.07	-0.10	-0.01
Avg. Cell Delay (ns)	0.923	1.242	0.287
Avg. Path Delay (ns)	1.082	1.895	0.429

Layout Comparison: Placement Guidance Effect



default placer

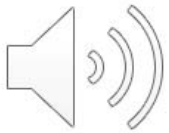


our approach

Discussions

- Observation 1: Fewer buffers are inserted
  - Reason: Grouping information is based on hierarchy
    - instances within a hierarchy tend to have more connections
  - Direct result: less wirelength (← less nets are created)

- Observation 2: Consistently achieve better timing
  - Reason: Macro affinity is captured by GNN
    - Logic-to-memory paths are usually the critical paths
    - Instances with similar affinity level will tend to be clustered together
      - Critical paths are balanced → better skew → better slack



Thank You for Listening! Q&A