



SHAPING THE NEXT GENERATION OF ELECTRONICS

JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA



JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

Automated layout symmetry annotation via graph node embedding

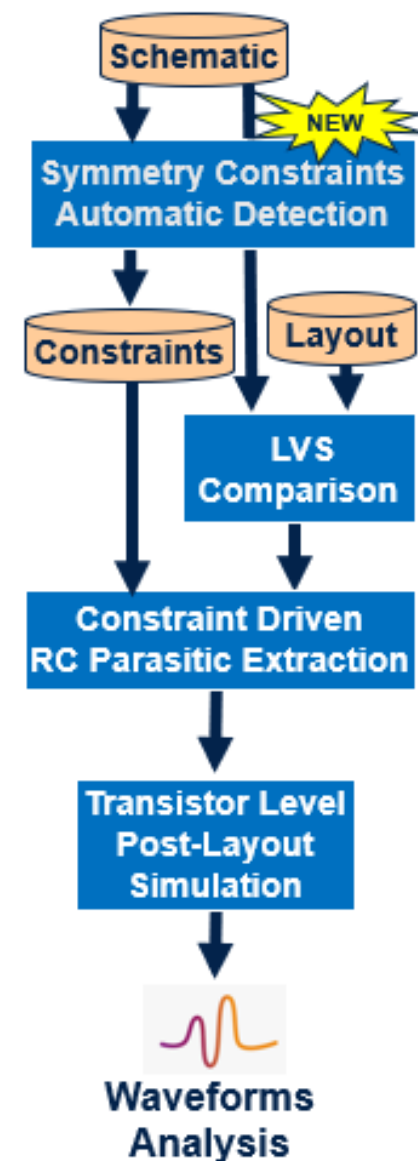
Jérôme Lescot, François Lemery
STMicroelectronics



Motivation

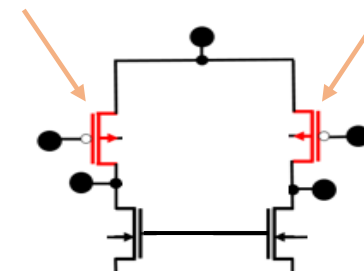
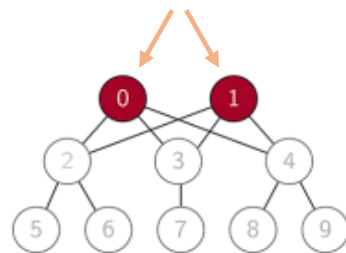
Automated constraint-driven analog layout flow

- Identification of analog layout **constraints** such as **symmetry**, & **matching** is a crucial task in advanced process nodes where parasitic effects can impact circuit reliability and lifetime
- The manual **annotation** of symmetry **constraints** requires analog design **expertise** and is error prone
- In this work, we propose an **unsupervised node embedding** method on a circuit netlist **graph** to capture **structural similarities** between nodes
- Applications:
 - Post-layout simulation (PLS) [1]
 - Analog constraint checking (ACC) [2]



Learning structural embeddings

- Our solution combines the advantages of an **unsupervised** diffusion-wavelet-method named *GraphWave* [3] applied to an accurate representation of the circuit, i.e. a **weighted undirected** and **bipartite graph**
- *GraphWave* [3] provides the mathematical proof that nodes of the graph with **similar embeddings** have **similar structural roles** in the graph, such as **symmetrical devices** in a circuit netlist

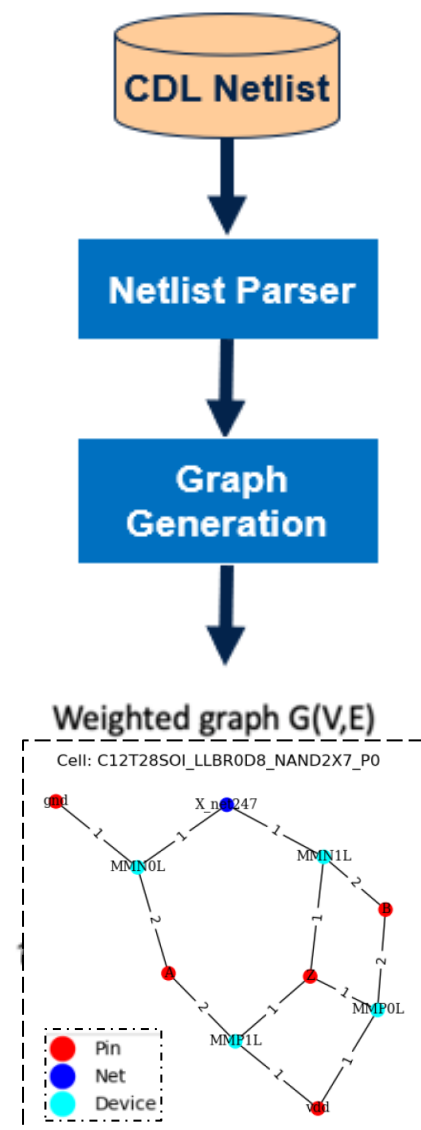
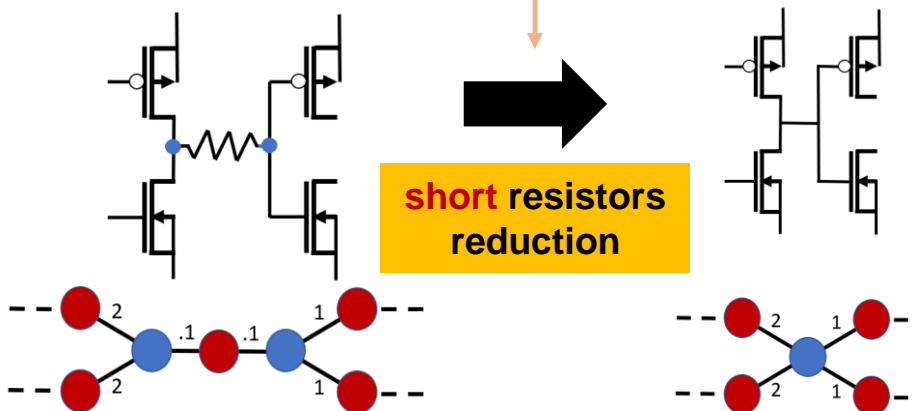
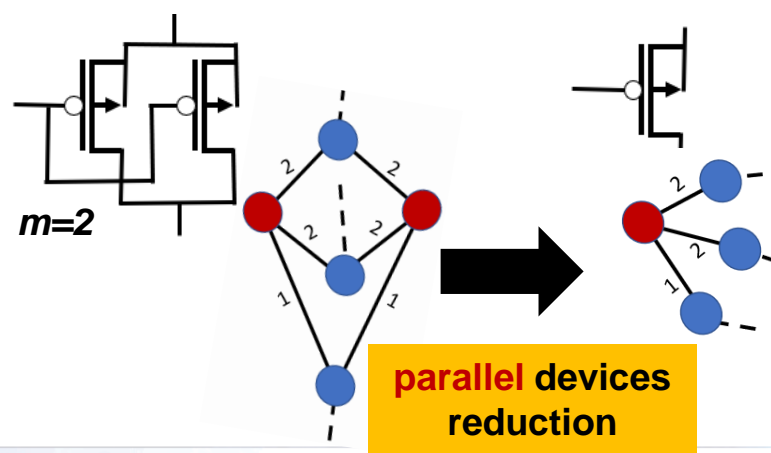


- Compared to **deep learning algorithms** applied on circuit graphs [4], this unsupervised method does not need datasets including well annotated netlists

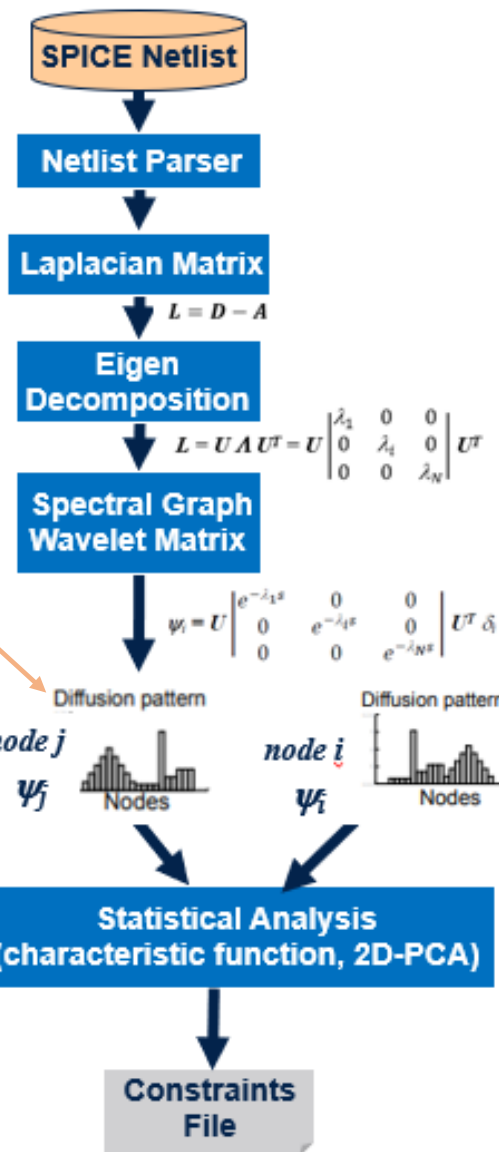
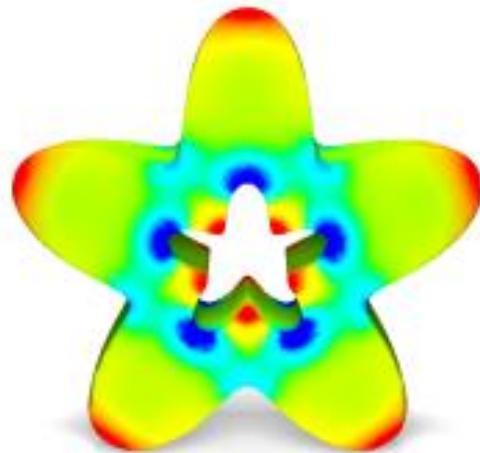
To our knowledge, this is the very **first work** that implements *GraphWave* node embeddings on **circuit netlists** in the context of **electronic design automation**

Circuit representation as a weighted bipartite graph

- **Netlist parser** implemented in Python OO language
 - Fast yet robust hierarchical parser validated on large SoCs netlists
 - Parallel reduction of devices that preserves equivalent parameters
- **Circuit graph** generated as Python [networkX](#) object
 - **Bipartite graph** including both **devices** and **electrical nodes** as graph vertices
 - Possibility to **ignore bulk terminal** of MOSFET devices
 - The **weight** of the edge represents the **type of the terminal** (e. g. *D*, *G*, *S*, *B*, ...)
 - Possibility to **reduce nodes of the graph** when detecting **short** resistors



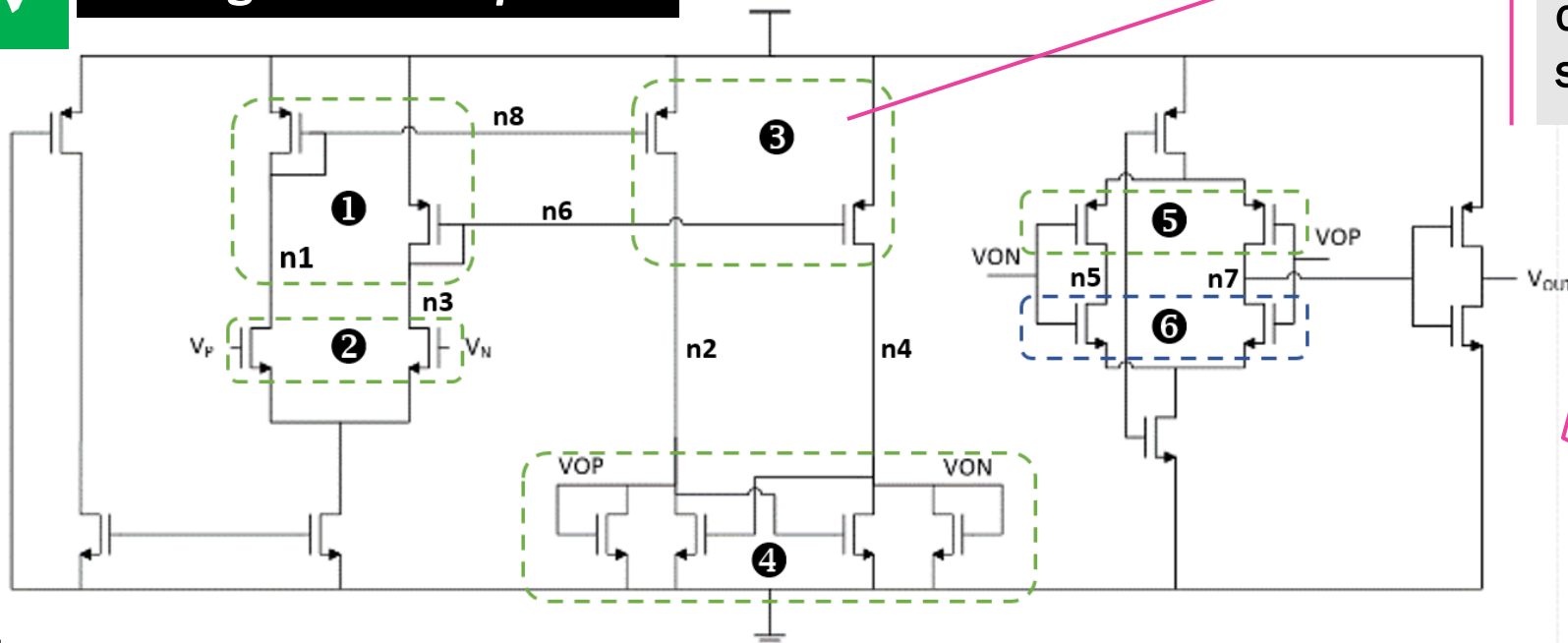
GraphWave implementation details



Experimental results



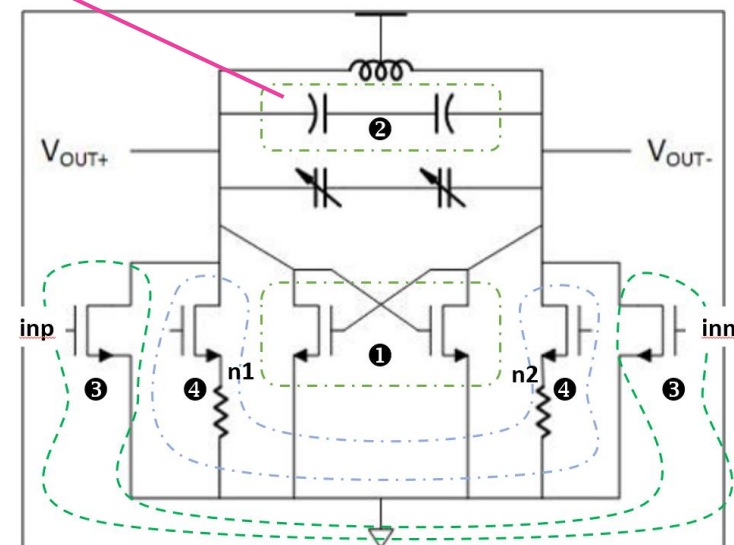
Analog circuit: *comparator*



✓ All symmetrical devices and nets are captured in various in-house netlists and some other circuits of public domain [5]

Symmetrical passive components can also be detected successfully

Analog circuit: *oscillator*



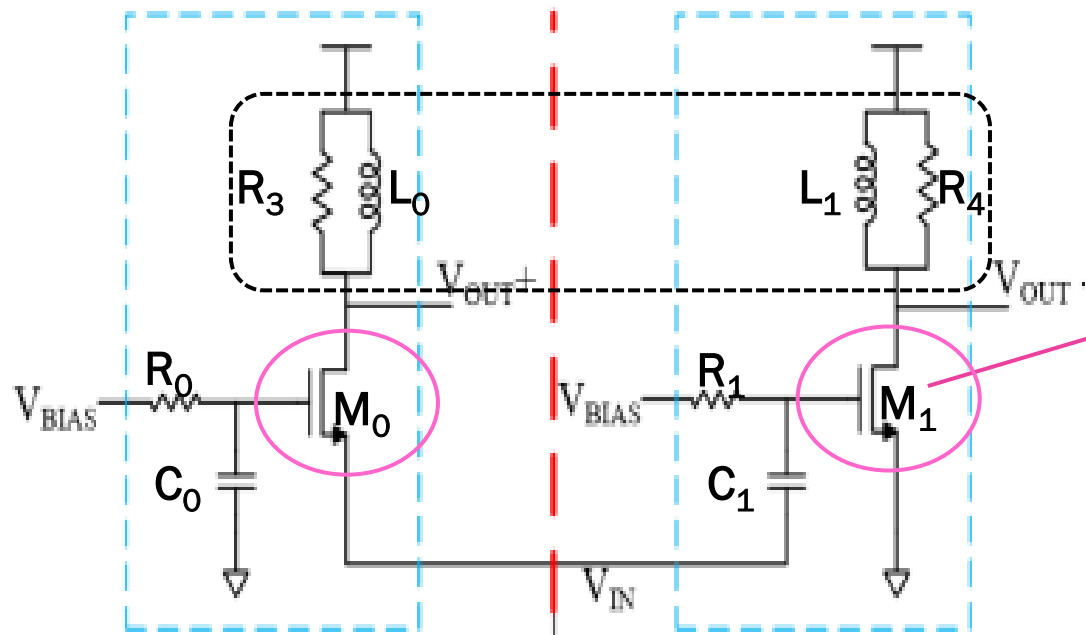
Capacity vs runtime (*)

Circuit	# Nodes	# Edges	Runtime (s)
PLL IP (18nm)	4803	10423	0.4
DoT IC (0.13μm)	897280	1988758	<10s

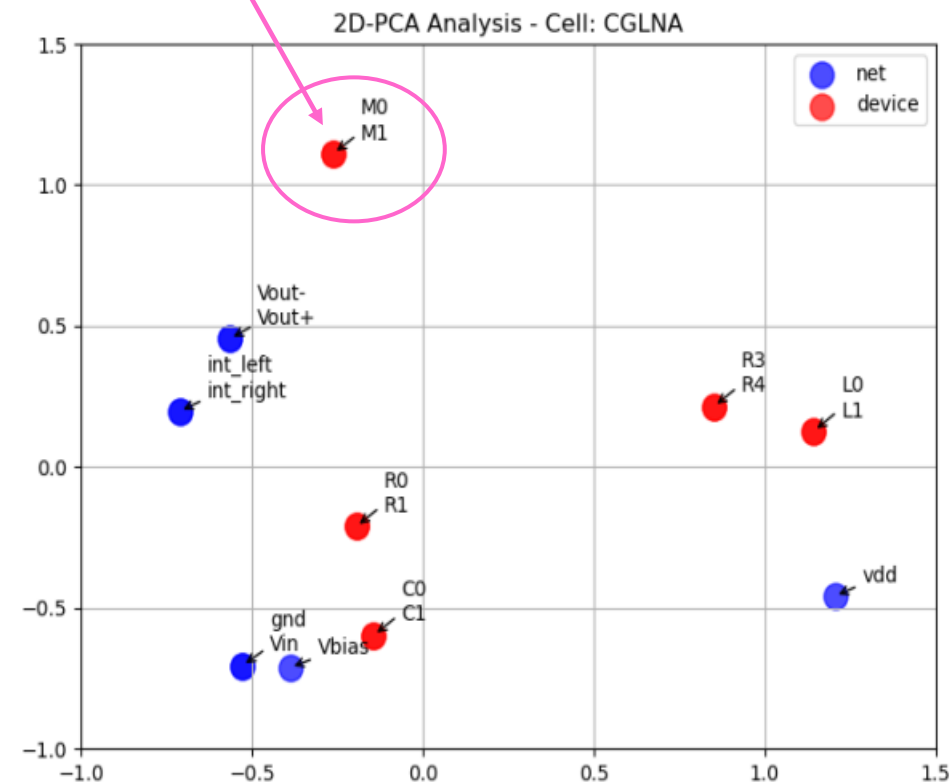
(*) Regular Linux RH8 machine on ST compute farm (no extra option)

Tolerance to graph perturbations

RF circuit: *low noise amplifier (LNA)*



✓ This technique allows for minor changes in circuit topology and can also detect *pseudo* symmetrical devices and nets



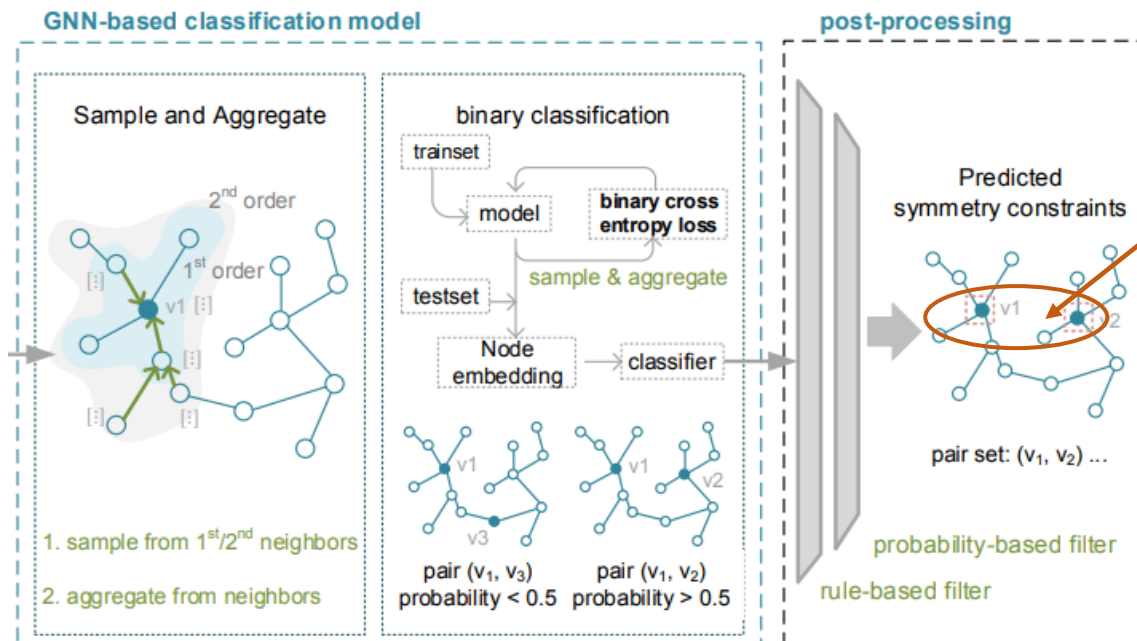
What about supervised ML solutions?

There exist attempts in literature [4] to achieve layout symmetry annotation with graph neural networks

- By using most popular GNNs such as *GraphSage* [6]
- Each node of the graph is encoded including a *feature* vector with *type* of the node, *distance* to p/g pins, etc..
- In every layer of the network, the algorithm aggregates node representation of neighbors with each node [6]

Main limitations of GNN solutions

- The **depth of the network** (K layers) determines the **maximum distance** from node to visited nodes in the graph
- Symmetry constraints are limited to **two devices only**
- Supervised methods require **datasets** including well annotated netlists (manual labels)
- The data are extremely **imbalanced** as device pairs with symmetry constraints represent a **small percentage** of all **device pairs**



GraphSage-based symmetry annotation flow [4]

Datasets	Circuits	Nodes	Edges	Valid pairs	Pos/Neg
S^3 -leaf	10	1378	9149	1522	89/1433
ALIGN-leaf	5	580	2134	576	48/528
OTA	5	684	3422	750	45/705

See table 2 in [4]

Conclusion

This **unsupervised structural node embedding method** can quickly capture symmetrical devices and nets in a circuit netlist by using a mathematically proven approach

- This **solution does not need any prior knowledge** such as datasets with annotated netlists
- All **experiments** conducted so far **validated** this new in-house solution
- This work brings evidence that *GraphWave* can be extended to **weighted** graphs
- This approach supports an **error tolerance** to small perturbations of the circuit graph to capture **almost**-symmetrical structures designed in actual **analog and RF** circuits

References

- [1] “Parasitic extraction for next-generation custom IC design”, white paper, [Synopsys®](#)
- [2] “Automated constraint checks enhance analog design reliability”, white paper, [Siemens-EDA](#)
- [3] Claire Donnat, Marinka Zitnik, David Hallac, Jure Leskovec, “Learning Structural Node Embeddings via Diffusion Wavelets,” KDD '18, August 19–23, 2018, London, United Kingdom
- [4] X. Gao, C. Deng, M. Liu, Z. Zhang, D. Z. Pan, and Y. Lin, “Layout symmetry annotation for analog circuits with graph neural networks,” in Proceedings of the Asia-South Pacific Design Automation Conference, 2021, pp. 152–157
- [5] K. Kunal, M. Madhusudan, A. K. Sharma, W. Xu, S. M. Burns, R. Harjani, J. Hu, D. A. Kirkpatrick, and S. S. Sapatnekar, “Align: Open-source analog layout automation from the ground up,” in Proc. DAC, 2019, pp. 1-4.
- [6] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in Proc. NeurIPS, 2017, pp. 1024–1034



SHAPING THE NEXT GENERATION OF ELECTRONICS

JUNE 23-27, 2024

MOSCONE WEST CENTER
SAN FRANCISCO, CA, USA

Thank You!

