A Fast Optimum Double Row Legalization Algorithm

Stefan Hougardy [1] **Meike Neuwohner** [1] Ulrike Schorr [2]

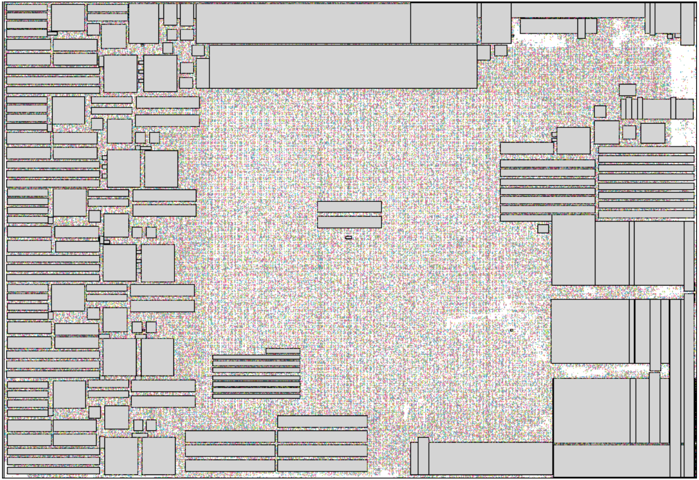
[1]University of Bonn

[2]Cadence Design Systems Inc.

Overview

- Introduction to the Placement Problem in Chip Design
- Approaches for Standard-Cell Legalization
- Challenge: double-row height cells
- Previous works on Multi-Row Legalization
- Our contribution: Optimizing squared cell movement in $\mathcal{O}(n \cdot \log(n))$ ($n :=$ # cells)
- Experimental results
- Summary

Placement Problem in Chip Design



Placement Flow

Step 1: Global Placement

Spread cells on chip area obeying local density constraints and approx. minimizing netlength.

Step 2: Legalization

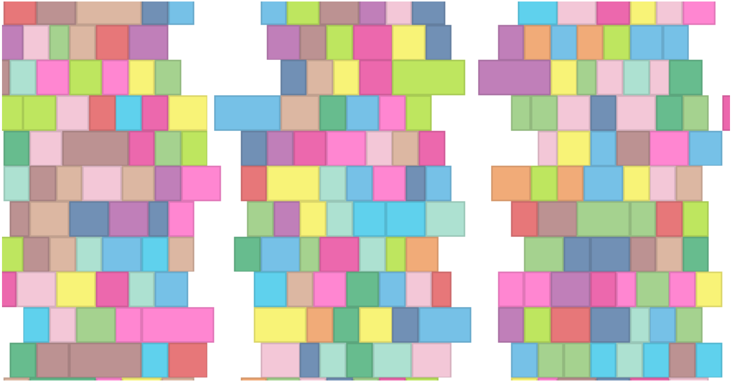
Move cells to overlap-free positions locally. Minimize cell movement or netlength.

Step 3: Detailed Placement

Post-Optimization

Cell Rows

Cells need to be placed in *cell rows*.



Approaches for Standard-Cell Legalization

Tetris [Hill, 2002], Abacus [Spindler et al., 2008]

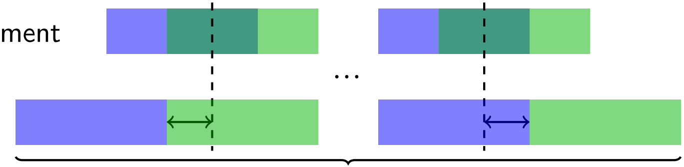
- Process cells from left to right (w.r.t. Global Placement positions).
- Greedily assign each cell to the "best" position in a nearby row.
    - closest free position (Tetris)
    - minimum cell movement (Abacus)

BonnTools [Brenner, 2013]

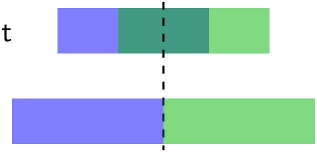
- Assign cells to unblocked parts of cell rows via a min-cost-flow-approach.
- Fix the left-to-right ordering within each row.
- Minimize squared cell movement.

Why *squared* cell movement?

- honor quality of Global Placement, e.g. timing properties
- balance movement



minimum L1 movement



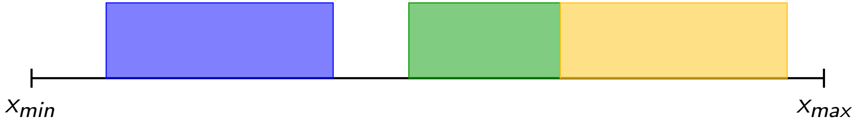
minimum squared cell movement

Single Row Problem

Definition (Single Row Problem)

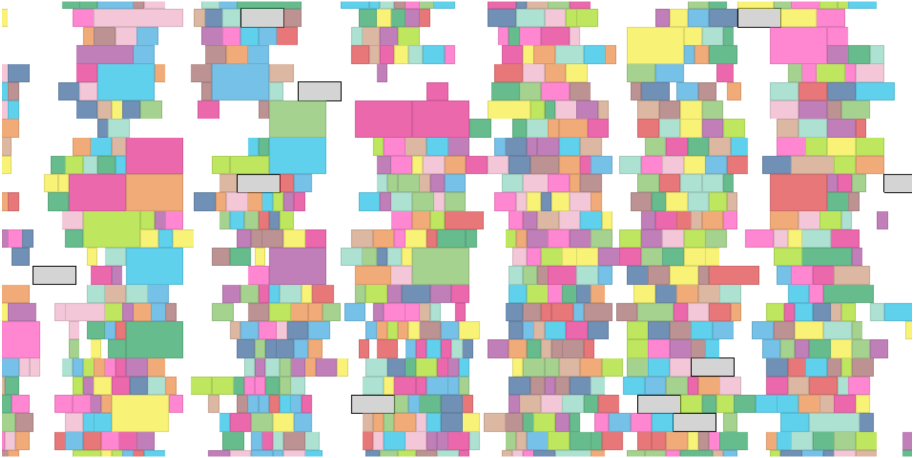
Input: Ordered set of cells in a row segment.

Task: Find overlap-free positions minimizing (squared) cell movement or netlength.



- solvable via the Clumping Algorithm [Kahng et al., 1999]
- linear runtime for squared cell movement
- n cells, m nets: $\mathcal{O}((n+m)\cdot\log(n))$ [Suhl, 2010]

Challenge: Cells of double-row height



⚠ The rows are not independent anymore!

How to handle cells of double-/multiple-row height?

- inflate or merge cells of single-row height to ensure uniform heights again [Wu and Chu, 2015]
- dynamic programming with restricted cell movement [Cheng et al., 2018], [Han et al., 2017], [Han et al., 2019], [Lin et al., 2016]
- solve a linear complementarity problem (LCP) [Chen et al., 2017], [Li et al., 2019], [Zhu et al., 2018]
- legalize small windows via integer linear programming [Hung et al., 2017]
- iterative cell insertion [Chow et al., 2016], [Li et al., 2018]
- try to adapt the Clumping Algorithm (*no guarantee* to find an optimum solution) [Wang et al., 2017]

How to handle cells of double-/multiple-row height?

- inflate or merge cells of single-row height to ensure uniform heights again [Wu and Chu, 2015]
- dynamic programming with restricted cell movement [Cheng et al., 2018], [Han et al., 2017], [Han et al., 2019], [Lin et al., 2016]
- solve a linear complementarity problem (LCP) [Chen et al., 2017], [Li et al., 2019], [Zhu et al., 2018]
- legalize small windows via integer linear programming [Hung et al., 2017]
- iterative cell insertion [Chow et al., 2016], [Li et al., 2018]
- try to adapt the Clumping Algorithm (*no guarantee* to find an optimum solution) [Wang et al., 2017]

How to handle cells of double-/multiple-row height?

- inflate or merge cells of single-row height to ensure uniform heights again [Wu and Chu, 2015]
- dynamic programming with restricted cell movement [Cheng et al., 2018], [Han et al., 2017], [Han et al., 2019], [Lin et al., 2016]
- solve a linear complementarity problem (LCP) [Chen et al., 2017], [Li et al., 2019], [Zhu et al., 2018]
- legalize small windows via integer linear programming [Hung et al., 2017]
- iterative cell insertion [Chow et al., 2016], [Li et al., 2018]
- try to adapt the Clumping Algorithm (*no guarantee* to find an optimum solution) [Wang et al., 2017]

How to handle cells of double-/multiple-row height?

- inflate or merge cells of single-row height to ensure uniform heights again [Wu and Chu, 2015]
- dynamic programming with restricted cell movement [Cheng et al., 2018], [Han et al., 2017], [Han et al., 2019], [Lin et al., 2016]
- solve a linear complementarity problem (LCP) [Chen et al., 2017], [Li et al., 2019], [Zhu et al., 2018]
- legalize small windows via integer linear programming [Hung et al., 2017]
- iterative cell insertion [Chow et al., 2016], [Li et al., 2018]
- try to adapt the Clumping Algorithm (*no guarantee* to find an optimum solution) [Wang et al., 2017]

How to handle cells of double-/multiple-row height?

- inflate or merge cells of single-row height to ensure uniform heights again [Wu and Chu, 2015]
- dynamic programming with restricted cell movement [Cheng et al., 2018], [Han et al., 2017], [Han et al., 2019], [Lin et al., 2016]
- solve a linear complementarity problem (LCP) [Chen et al., 2017], [Li et al., 2019], [Zhu et al., 2018]
- legalize small windows via integer linear programming [Hung et al., 2017]
- iterative cell insertion [Chow et al., 2016], [Li et al., 2018]
- try to adapt the Clumping Algorithm (*no guarantee* to find an optimum solution) [Wang et al., 2017]

How to handle cells of double-/multiple-row height?

- inflate or merge cells of single-row height to ensure uniform heights again [Wu and Chu, 2015]
- dynamic programming with restricted cell movement [Cheng et al., 2018], [Han et al., 2017], [Han et al., 2019], [Lin et al., 2016]
- solve a linear complementarity problem (LCP) [Chen et al., 2017], [Li et al., 2019], [Zhu et al., 2018]
- legalize small windows via integer linear programming [Hung et al., 2017]
- iterative cell insertion [Chow et al., 2016], [Li et al., 2018]
- try to adapt the Clumping Algorithm (*no guarantee* to find an optimum solution) [Wang et al., 2017]

How to handle cells of double-/multiple-row height?

- inflate or merge cells of single-row height to ensure uniform heights again [Wu and Chu, 2015]
- dynamic programming with restricted cell movement [Cheng et al., 2018], [Han et al., 2017], [Han et al., 2019], [Lin et al., 2016]
- solve a linear complementarity problem (LCP) [Chen et al., 2017], [Li et al., 2019], [Zhu et al., 2018]
- legalize small windows via integer linear programming [Hung et al., 2017]
- iterative cell insertion [Chow et al., 2016], [Li et al., 2018]
- try to adapt the Clumping Algorithm (*no guarantee* to find an optimum solution) [Wang et al., 2017]

Our contribution

Minimize squared cell movements for two adjacent rows containing cells of single- and double-row height in $\mathcal{O}(n \cdot \log(n))$ ($n := \#$ cells).

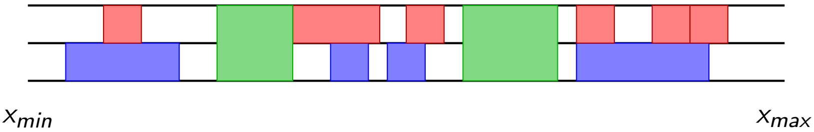
The Double Row Problem

Double Row Problem

Input: cells of single- and double-row cells to be placed in two adjacent rows

- fixed assignment to rows
- fixed left-to-right ordering

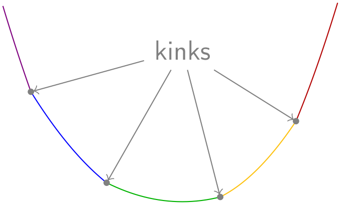
Task: Minimize squared cell movement.



Solving the Double Row Problem

Idea

Reduce to an instance of the Single Row Problem with *piecewise quadratic* cost functions on the set of double row cells.



A piecewise quadratic function.

Implementation of the Clumping Algorithm with piecewise quadratic cost functions

- adapt implementation for piecewise linear cost functions by [Suhl, 2010]
- runs in $\mathcal{O}((m+n)\cdot\log(\min\{n,m\}))$ ($n :=$ # cells, $m :=$ #kinks)

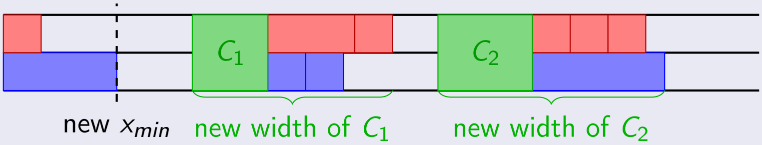
Reducing the Double Row Problem to the Single Row Problem

Idea

Reduce to a Single Row Instance on the double row cells.
→ Determine *feasibility* and *costs* depending on their positions.

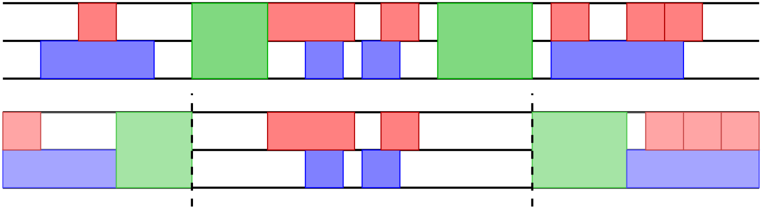
Feasibility

Increase widths/ x_{min} to ensure the single row cells fit in between.

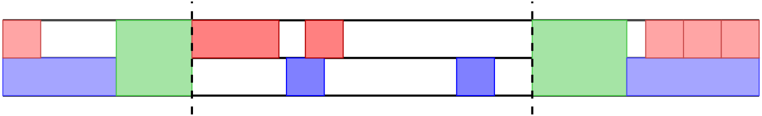


Optimum positions for the single row cells depending on the positions of the double row cells

Define upper and lower subinstance for the *i*-th gap:

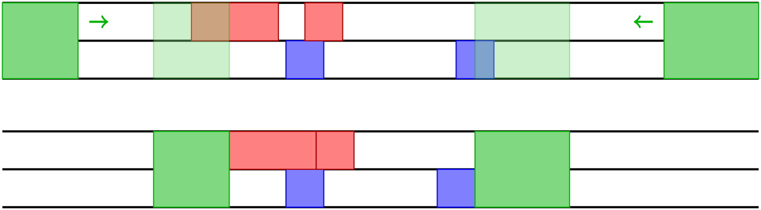


and solve them optimally.

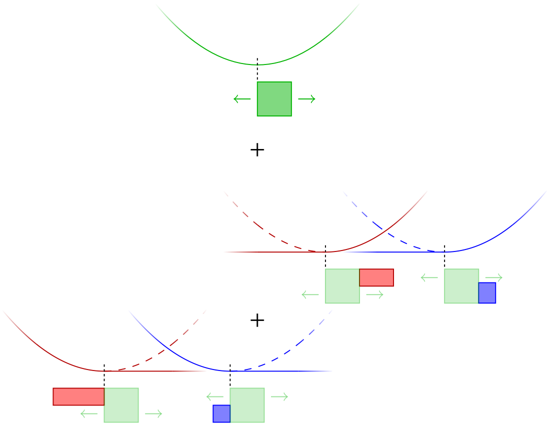


Optimum positions for the single row cells depending on the positions of the double row cells

Move the single row cells as few as possible to fit into the respective gap.



New objective for a double row cell



own objective

costs for moving cells
in the right gap
further to the right

costs for moving cells
in the left gap
further to the left

⇒ convex, piecewise quadratic objective function

Solving the Double Row Problem

Theorem

The Double Row Problem with quadratic cost functions and n *cells can be solved in time* $\mathcal{O}(n \cdot \log(n))$.

Experimental results

Our Legalization Flow

1. Assign cells to unblocked parts of circuit rows and fix the left-to-right ordering as in [Brenner, 2013]
    - Optimizes squared cell movement.
2. Optimize cell movement for pairs of adjacent rows.

Experiment 1

Compare with: [Chen et al., 2017] , [Li et al., 2019] and [Brenner, 2013]

Benchmarks: from the ICCAD-2017 CAD Contest on Multi-Deck Standard-Cell Legalization [Darav et al., 2017]

- omit fence region constraints and soft constraints
- respect power-alignment constraints

Objective: L1 cell movement (not the primary purpose of our algorithm!)

Goal: establish competitiveness of our approach

Experiment 1

Instance	Av. L1 Movement (Sites)				
	[Chen et al., 2017]	[Li et al., 2019]	[Brenner, 2013]	Ours	$\frac{\text{Ours}}{\text{[Li et al., 2019]}}$
des_perf_1	10.86	6.97	6.66	6.66	95.55%
des_perf_ a_md1	6.71	5.94	5.85	5.79	97.47%
des_perf_a_md2	6.77	5.93	6.08	6.07	102.36%
des_perf_b_md1	5.17	4.77	4.78	4.72	98.95%
des_perf_b_md2	5.74	5.25	5.38	5.31	101.14%
edit_dist_1_md1	6.22	5.79	5.75	5.69	98.27%
edit_dist_a_md2	6.02	5.51	5.57	5.51	100.00%
edit_dis_ a_md3	9.11	7.08	6.96	6.93	97.88%
fft_2_md2	8.84	7.54	7.89	7.76	102.92%
fft_a_md2	5.03	4.86	4.74	4.70	96.71%
ff_ a_md3	4.73	4.55	4.43	4.42	97.14%
pci_bridge32_a_md1	6.01	5.64	5.83	5.76	102.13%
pci_bridge32_a_md2	9.43	7.14	7.55	7.45	104.34%
pc_ bridge32_b_md1	6.35	6.01	5.79	5.72	95.17%
pci_bridge32_b_md2	5.92	5.53	5.43	5.42	98.01%
pci_bridge32_b_md3	6.74	6.10	6.13	6.12	100.33%
average	6.85	5.91	5.93	5.88	99.27%

Experiment 2

Compare with: [Brenner, 2013] (greedily legalizes and fixes cells of double-row height in advance)

Benchmarks: from the ISPD 2015 Detailed Routing-Driven Placement Contest [Bustany et al., 2015]

- modified by [Chow, Pui, and Young, 2016] to contain a significant amount of cells of double-row height

Objective: squared cell movement

Goal: examine effectiveness of our approach to reduce squared cell movement

Experiment 2

Instance	fraction of double row cells	Squared Cell Movement		
		[Brenner, 2013]	Ours	$\frac{\text{Ours}}{\text{[Brenner, 2013]}}$
des_perf_1	7.81%	4.15E+10	2.82E+10	68.00%
des_perf_a	7.86%	3.66E+09	2.51E+09	68.53%
des_perf_b	7.81%	3.84E+09	2.49E+09	64.94%
edit_dist_a	4.32%	4.49E+09	3.17E+09	70.55%
fft_1	6.15%	9.53E+09	5.54E+09	58.18%
fft_2	6.15%	1.95E+09	1.20E+09	61.43%
fft_a	6.23%	1.30E+09	9.04E+08	69.41%
fft_b	6.23%	1.89E+09	1.27E+09	67.13%
matrix_mult_1	1.87%	9.80E+09	6.81E+09	69.47%
matrix_mult_2	1.87%	8.26E+09	5.68E+09	68.77%
matrix_mult_a	1.88%	2.97E+09	2.31E+09	77.89%
matrix_mult_b	1.87%	2.61E+09	2.15E+09	82.07%
pci_bridge32_a	11.01%	1.23E+09	7.93E+08	64.60%
pci_bridge32_b	11.00%	6.13E+08	3.61E+08	58.86%
superblue11_a	6.95%	2.67E+11	2.48E+11	92.74%
superblue12	8.89%	5.61E+11	5.38E+11	95.84%
superblue14	7.75%	2.00E+11	1.81E+11	90.19%
superblue16 a	8.09%	6.35E+10	4.76E+10	74.99%
superblue19	5.53%	1.22E+11	1.14E+11	93.21%
average				73.51%

Summary

- Cells of double- or even arbitrary multiple-row height create new challenges in legalization.

Summary

- Cells of double- or even arbitrary multiple-row height create new challenges in legalization.
- We have shown how to minimize squared cell movement for pairs of adjacent cell rows in the presence of cells of double-row height.

Summary

- Cells of double- or even arbitrary multiple-row height create new challenges in legalization.
- We have shown how to minimize squared cell movement for pairs of adjacent cell rows in the presence of cells of double-row height.
- Without artificially bounding the maximum cell movement, we can ensure a running time of $\mathcal{O}(n \cdot \log(n))$ ($n := \#$ cells).

Summary

- Cells of double- or even arbitrary multiple-row height create new challenges in legalization.
- We have shown how to minimize squared cell movement for pairs of adjacent cell rows in the presence of cells of double-row height.
- Without artificially bounding the maximum cell movement, we can ensure a running time of $\mathcal{O}(n \cdot \log(n))$ ($n :=$ # cells).
- Even though not explicitly designed to do so, our algorithm can compete with state-of-the-art legalizers in minimizing L1 cell movement.

Summary

- Cells of double- or even arbitrary multiple-row height create new challenges in legalization.
- We have shown how to minimize squared cell movement for pairs of adjacent cell rows in the presence of cells of double-row height.
- Without artificially bounding the maximum cell movement, we can ensure a running time of $\mathcal{O}(n \cdot \log(n))$ ($n :=$ # cells).
- Even though not explicitly designed to do so, our algorithm can compete with state-of-the-art legalizers in minimizing L1 cell movement.
- Compared to a legalization approach where the double-row height cells are fixed in advance, we can significantly reduce squared cell movement.

References I

U. Brenner. BonnPlace Legalization: Minimizing Movement by Iterative Augmentation. *TCAD*, 32(8):1215–1227, 2013.

I. Bustany, D. Chinnery, J. Shinnerl, and V. Yutsis. ISPD 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement. In *Proceedings of the ISPD*, pages 157–164, 2015.

J. Chen, Z. Zhu, W. Zhu, and Y. Chang. Toward Optimal Legalization for Mixed-Cell-Height Circuit Designs. In *54th DAC*, 2017. ISBN 9781450349277.

Y. Cheng, D. Huang, W. Mak, and T. Wang. A Practical Detailed Placement Algorithm under Multi-Cell Spacing Constraints. In *Proceedings of the ICCAD*, 2018.

W. Chow, C. Pui, and E. Young. Legalization Algorithm for Multiple-Row Height Standard Cell Design. In *53rd DAC*, pages 1–6, 2016.

N. Darav, I. Bustany, A. Kennings, and R. Mamidi. ICCAD-2017 CAD Contest in Multi-Deck Standard Cell Legalization and Benchmarks. In *ICCAD*, pages 867–871, 2017.

C. Han, K. Han, A. Kahng, H. Lee, L. Wang, and B. Xu. Optimal Multi-Row Detailed Placement for Yield and Model-Hardware Correlation Improvements in Sub-10nm VLSI. In *ICCAD*, pages 667–674, 2017.

References II

C. Han, A. Kahng, L. Wang, and B. Xu. Enhanced Optimal Multi-Row Detailed Placement for Neighbor Diffusion Effect Mitigation in Sub-10 nm VLSI. *TCAD*, 38 (9):1703–1716, 2019.

D. Hill. Method and system for high speed detailed placement of cells within an integrated circuit design. U.S. Patent 6370673, 2002.

C. Hung, P. Chou, and W. Mak. Mixed-Cell-Height Standard Cell Placement Legalization. In *Proceedings of the Great Lakes Symposium on VLSI*, pages 149–154, 2017.

A. Kahng, P. Tucker, and A. Zelikovsky. Optimization of Linear Placements for Wirelength Minimization with Free Sites. In *Proceedings of the Asia and South Pacific Design Automation Conference*, pages 241–244, 1999.

H. Li, W. Chow, G. Chen, E. Young, and B. Yu. Routability-driven and fence-aware legalization for mixed-cell-height circuits. In *55th DAC*, pages 1–6, 2018.

X. Li, J. Chen, W. Zhu, and Y. Chang. Analytical mixed-cell-height legalization considering average and maximum movement minimization. In *Proceedings of the ISPD*, pages 27–34, 2019.

References III

Y. Lin, B. Yu, X. Xu, J. Gao, N. Viswanathan, W. Liu, Z. Li, C. Alpert, and D. Pan. MrDP: Multiple-row Detailed Placement of Heterogeneous-sized Cells for Advanced Nodes. In *ICCAD*, pages 1–8, 2016.

P. Spindler, U. Schlichtmann, and F. Johannes. Abacus: Fast Legalization of Standard Cell Circuits with Minimal Movement. In *Proceedings of the ISPD*, pages 47–53, 2008.

U. Suhl. Row-Placement in VLSI Design: The Clumping Algorithm and a generalization. diploma thesis, University of Bonn, Research Institute for Discrete Mathematics, April 2010.

C. Wang, Y. Wu, J. Chen, Y. Chang, S. Kuo, W. Zhu, and G. Fan. An Effective Legalization Algorithm for Mixed-Cell-Height Standard Cells. In *22nd Asia and South Pacific Design Automation Conference*, pages 450–455, 2017.

G. Wu and C. Chu. Detailed Placement Algorithm for VLSI Design with Double-Row Height Standard Cells. *TCAD*, 35:1569–1573, 2015.

Z. Zhu, X. Li, Y. Chen, J. Chen, W. Zhu, and Y. Chang. Mixed-Cell-Height Legalization Considering Technology and Region Constraints. In *Proceedings of the ICCAD*, 2018.

Thank you for watching!

In case you have questions or comments,
feel free to contact us!

`hougardy@or.uni-bonn.de`

`neuwohner@or.uni-bonn.de`

`uschorr@cadence.com`