EDA
Institute for Electronic
Design Automation
Prof. Ulf Schlichtmann

Abacus: Fast Legalization of Standard Cell Circuits with Minimal Movement

Peter Spindler, Ulf Schlichtmann and Frank M. Johannes
Technische Universitaet Muenchen

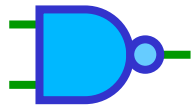
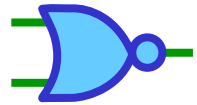
ISPD, April 2008

Outline

- Background
- State-of-the-Art
- Abacus, PlaceRow
- Experimental Results
- Conclusion

Background

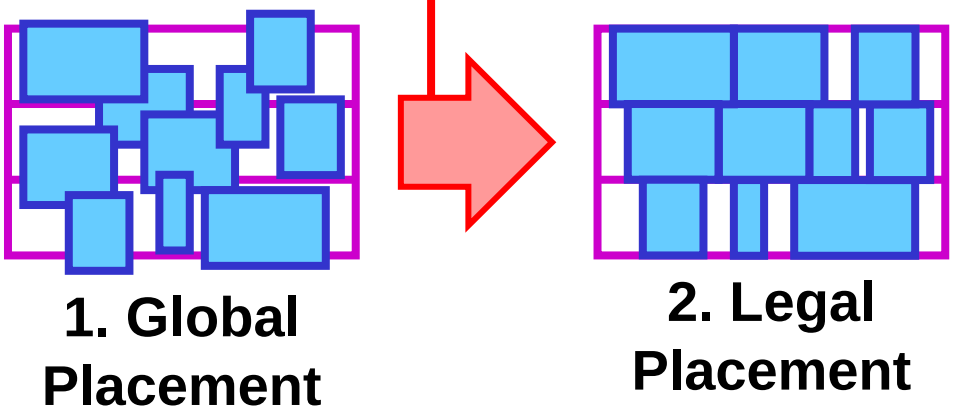
Standard cell circuits:

Millions of cells: inverter  , NAND  , NOR 

Physical representation:

Rectangles, all have the same height, but different widths 

Placement: Align cells overlap-free to row structure



Legal placement:

- No overlap
- Cells aligned to rows
- **Preserve global placement: minimal cell movement**

State-of-the-Art in Legalization

State-of-the-Art:

- **Flow based:** assign cells to places
  Domino [Doll et al., TCAD 1994]
  BonnPlace [Vygen et al., TCAD 2004]
- **Two stage:** first assign cells to rows, then place the rows
  [Madden et al., ICCAD 2003, Kahng et al., GLS-VLSI 2004]
- **Diffusion based** [Alpert et al., DAC 2005]
- **Computational Geometry based** [Alpert et al., DAC 2007]
- **Greedy:** legalize one cell at a time: Tetris [Hill, Patent, 2002]

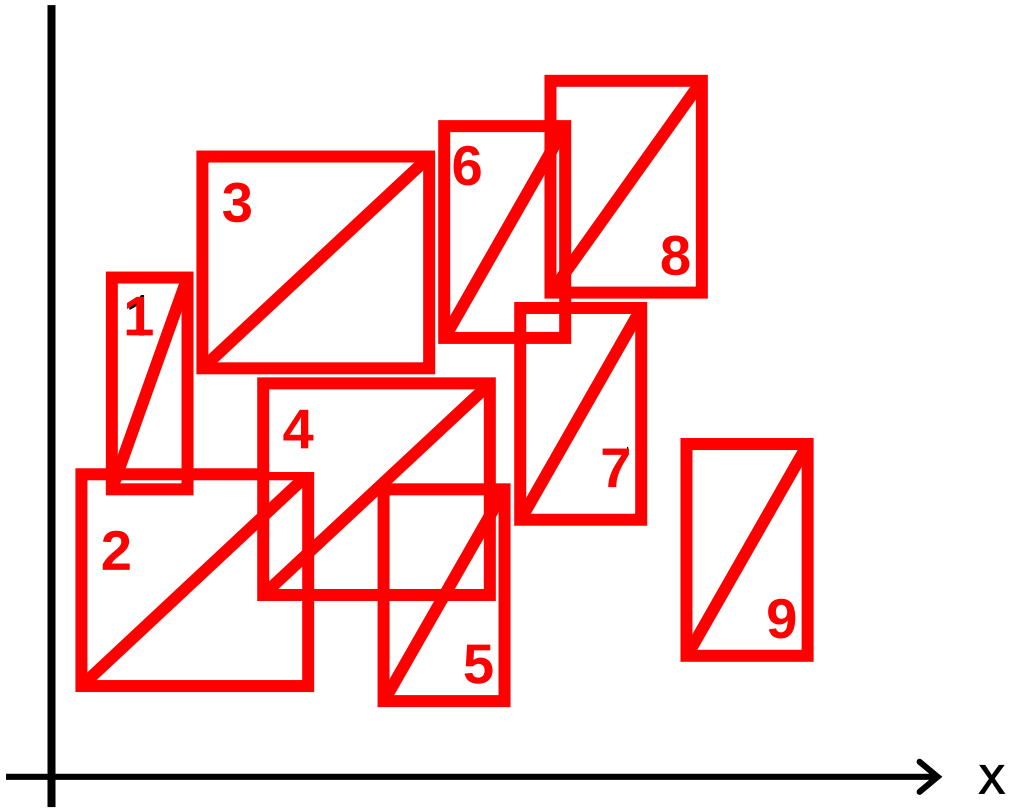
Abacus: Overview

Abacus:

- Similar to Tetris: sort cells, legalize one cell at a time
- Legalization of one cell: move cell over the rows, place cell to best/nearest row
- Difference to Tetris: **PlaceRow**: move already legalized cells within one row, minimize total movement
- Because of **PlaceRow**: lower total movement

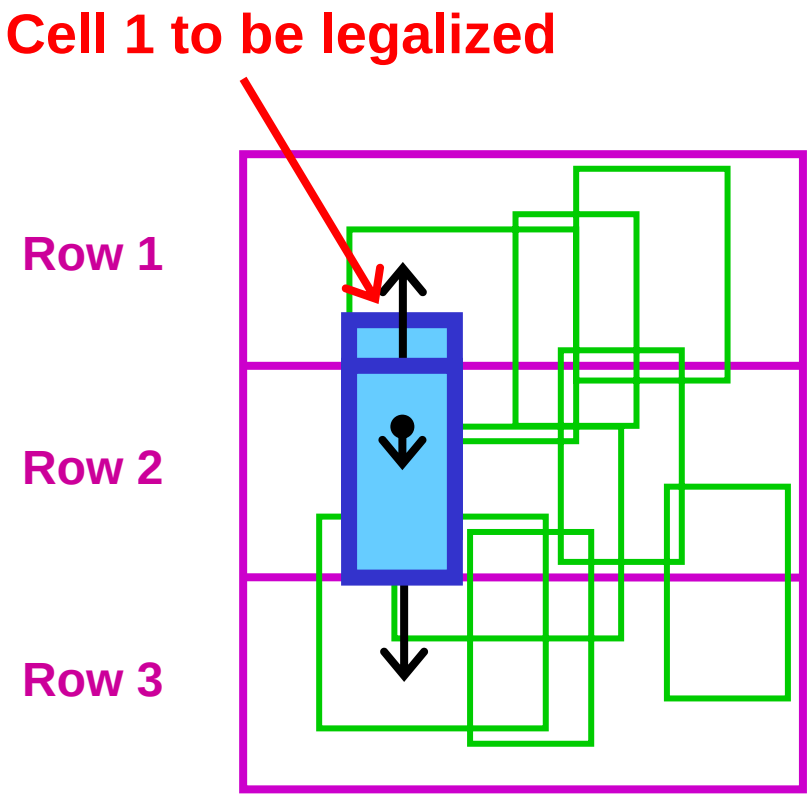
Step 1: Sorting

Sort cells according to x-pos in global placement



Process order: 1, 2, 3, 4, 5, 6, 7, 8, 9

Legalize Cell 1



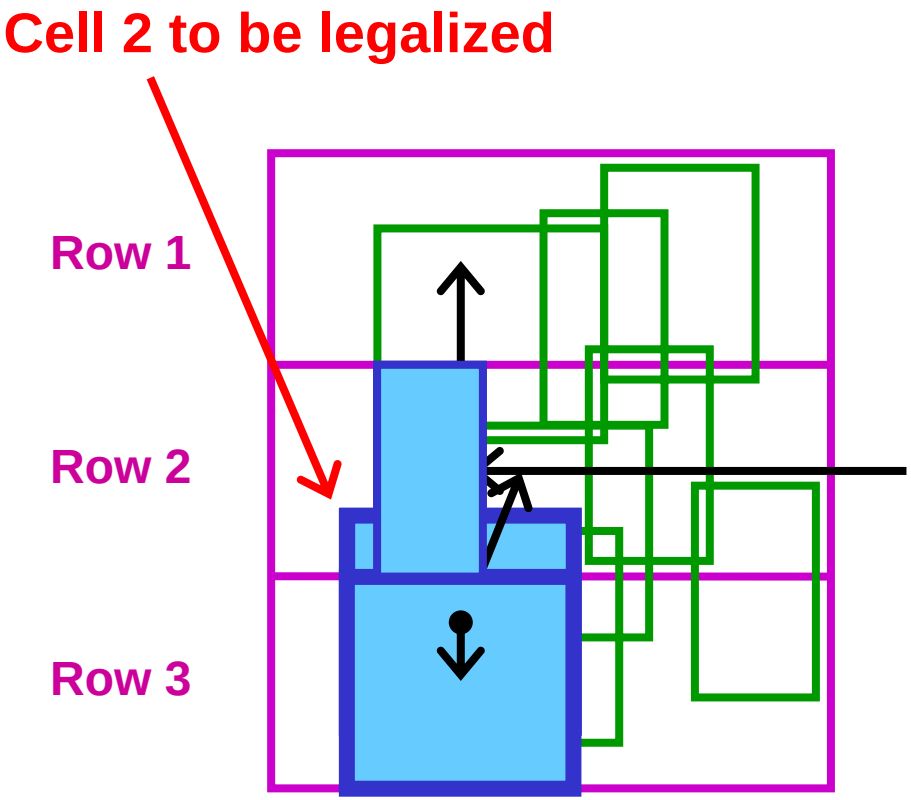
Insert to row 1

Insert to row 2

Insert to row 3

Best row: 2

Legalize Cell 2



Insert to row 1

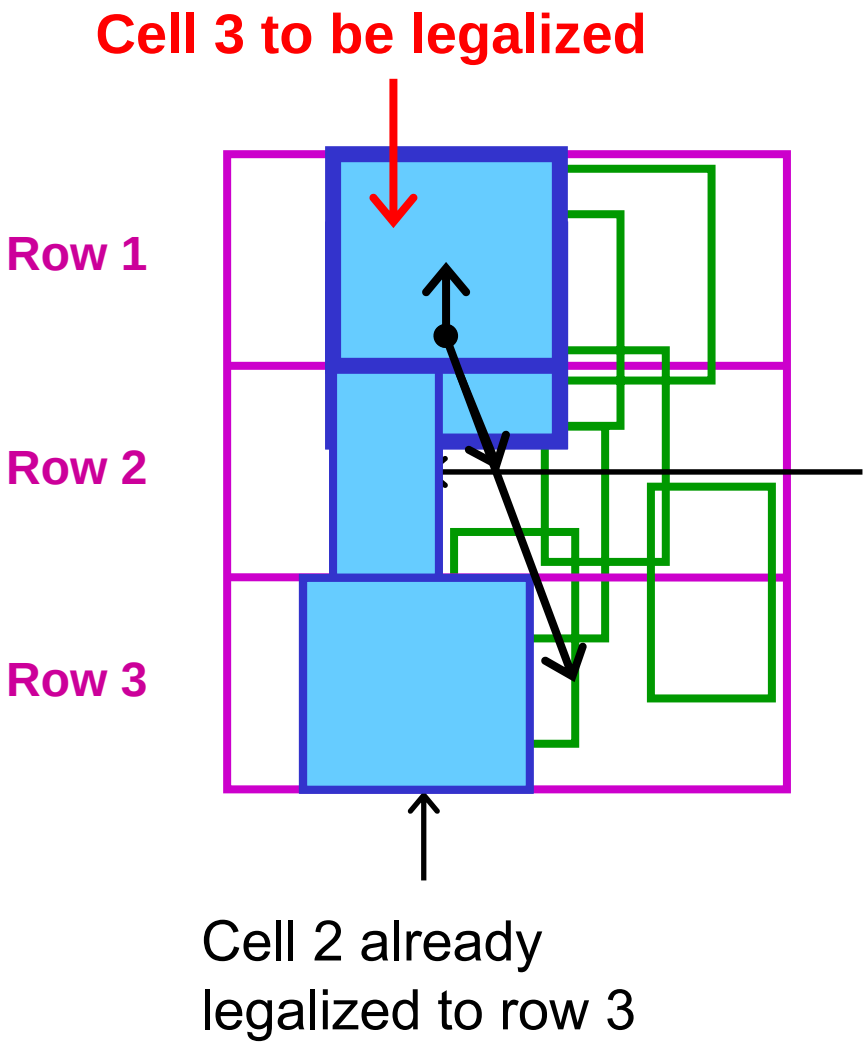
Insert to row 2

PlaceRow 2: Minimize quadratic movement in x-direction of cells

Insert to row 3

Best row: 3

Legalize Cell 3



Insert to row 1

Insert to row 2

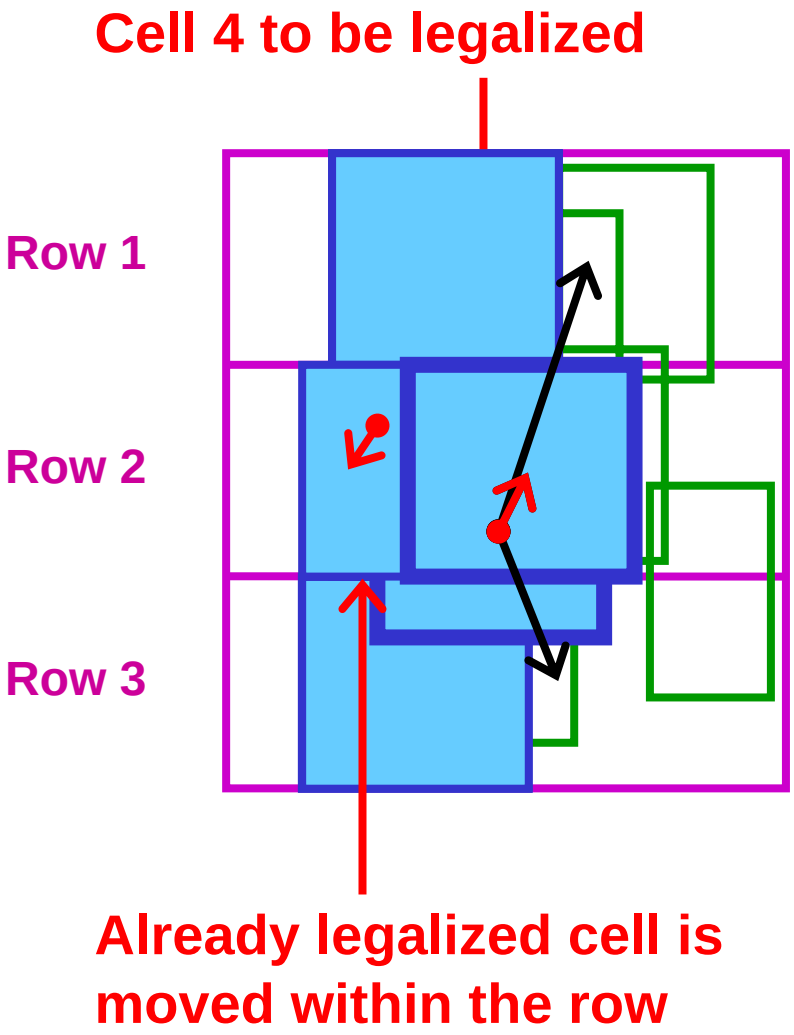
PlaceRow 2: Minimize quadratic movement in x-dir of cells

Insert to row 3

PlaceRow 3: Minimize quadratic movement in x-dir of cells

Best row: 1

Legalize Cell 4



Insert to row 1

PlaceRow 1

Insert to row 2

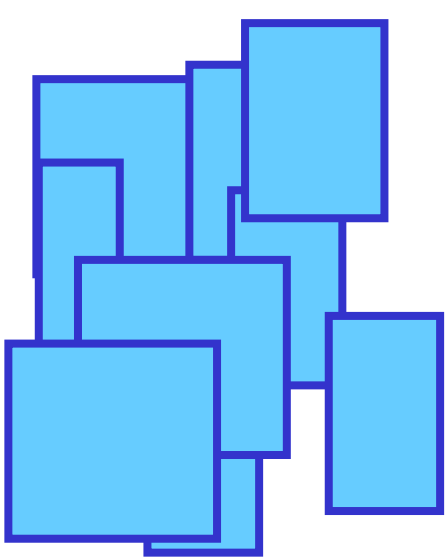
PlaceRow 2

Insert to row 3

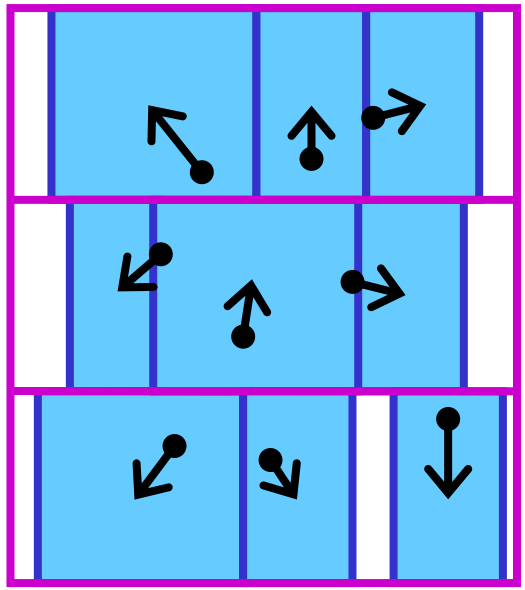
PlaceRow 3

Best row: 2

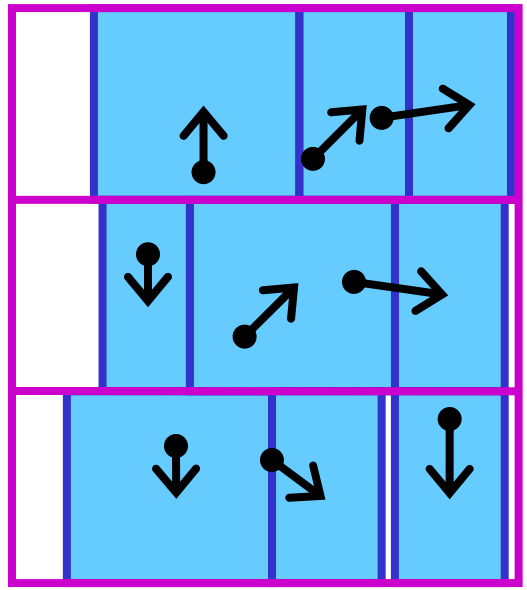
Final Result



Global Placement



Abacus



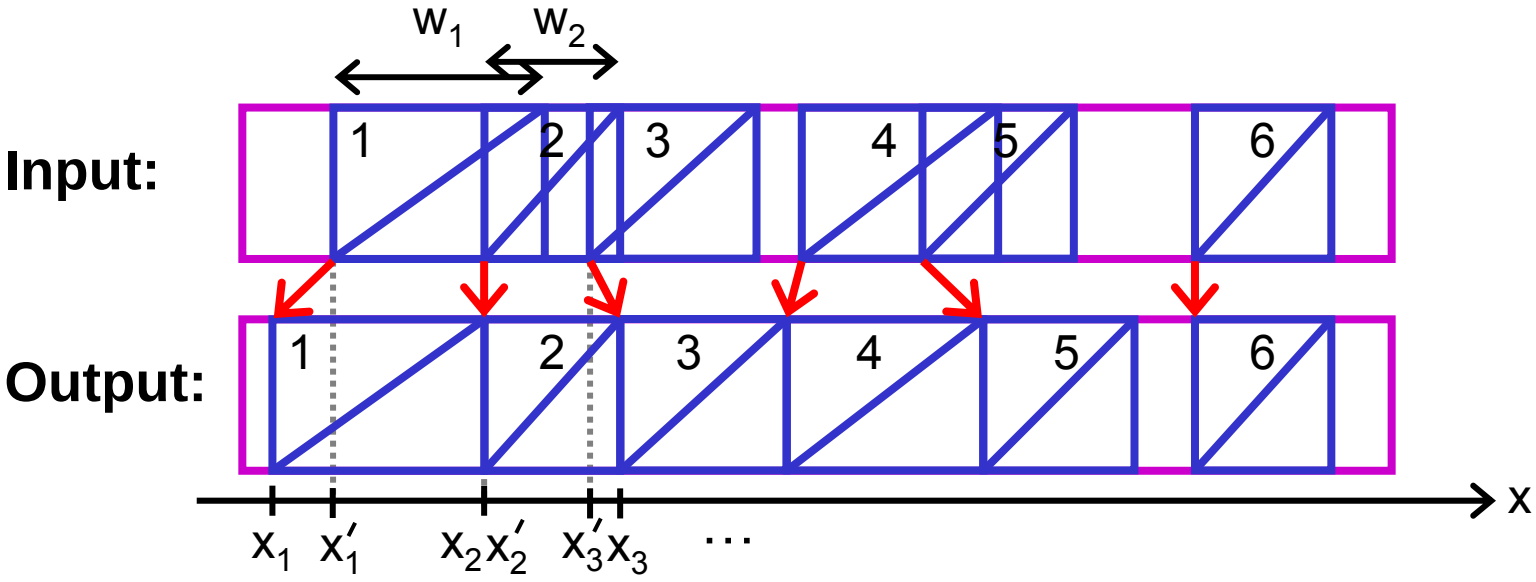
Tetris:

No PlaceRow → higher movement (about 50% more for a large circuit)

PlaceRow

Input: one row with N cells, x-pos of cells: global placement (x_i')

Output: new (legal) x-pos of cells (x_i) such that the overlap is removed and the total quadratic movement is minimized



QP:

$$\min \sum_{i=1}^{N} e_i \left(x_i - x_i'\right)^2 \quad \text{s.t.} \quad \underbrace{x_i \geq x_{i-1} + w_{i-1}}_{\text{no overlap}} \leftarrow \text{width}$$

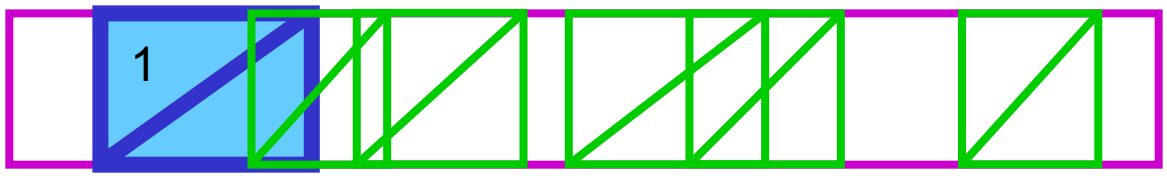
weight; legal x-pos; global x-pos

PlaceRow: Dynamic Programming

PlaceRow:

- Solve QP by dynamic programming approach:
  solve sub problems optimally to obtain final solution
- Process cells from left to right

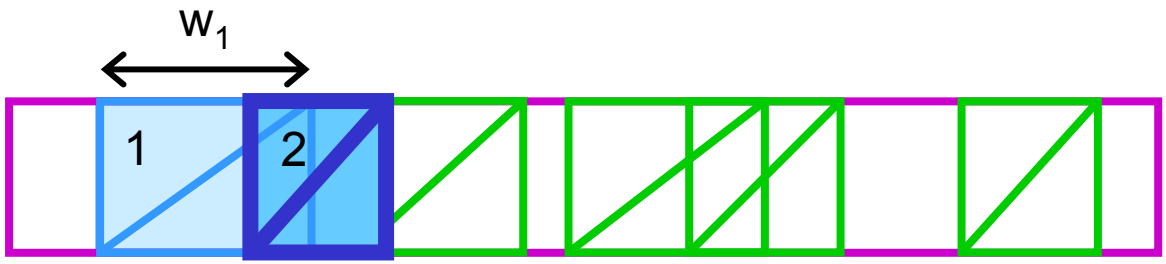
Cell 1:



first cell → do not move

PlaceRow, Cell 2

Cell 2:



overlap with previous cell? **yes** ➔ **cluster** with previous cell

Clustering process: $x_2 = x_1 + w_1$

$$\min e_1\left(x_1 - x_1'\right)^2 + e_2\left(x_2 - x_2'\right)^2 \Rightarrow \min\underbrace{\left(e_1 + e_2\right)}_{\text{new } e_1}\left(x_1 - \underbrace{\frac{e_1 x_1' + e_2\left(x_2' - w_1\right)}{e_1 + e_2}}_{\text{new } x_1'}\right)^2$$

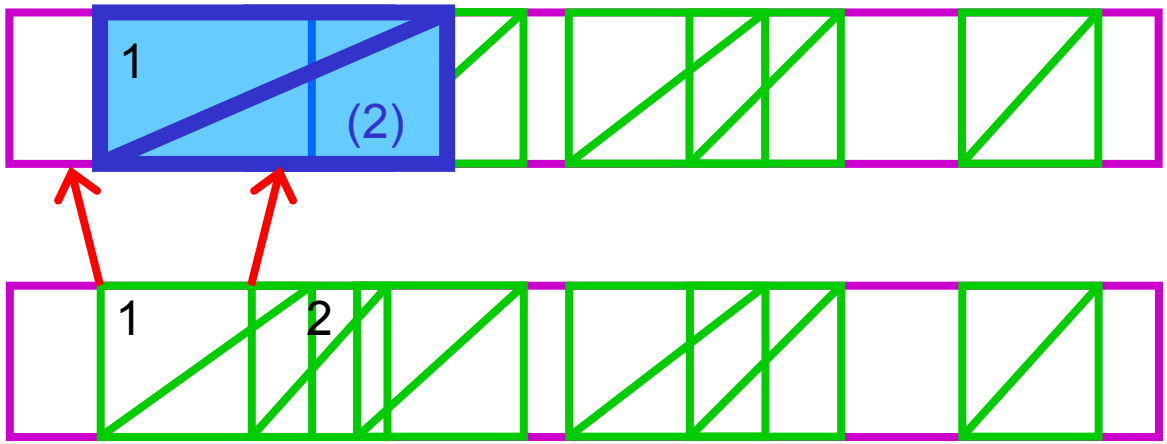
Update x_1', e_1, and w_1:

$$x_1' \leftarrow \frac{e_1 x_1' + e_2\left(x_2' - w_1\right)}{e_1 + e_2} \qquad e_1 \leftarrow e_1 + e_2 \qquad w_1 \leftarrow w_1 + w_2$$

Result: $\min e_1\left(x_1 - x_1'\right)^2$ ➔ $x_1 = x_1'$

PlaceRow, Cell 2 (Cont‘d)

Animation:

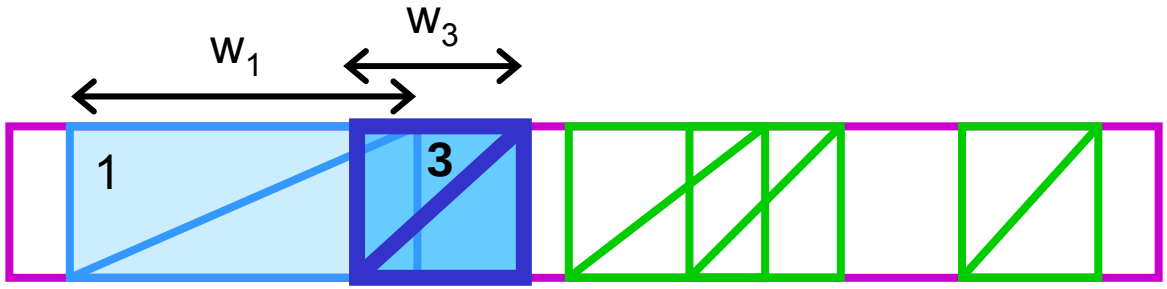


Cluster cell 1 and 2

Move cluster to new global x-pos

PlaceRow, Cell 3

Cell 3:



Overlap with previous cell? **yes** ➔ **cluster** with previous cell

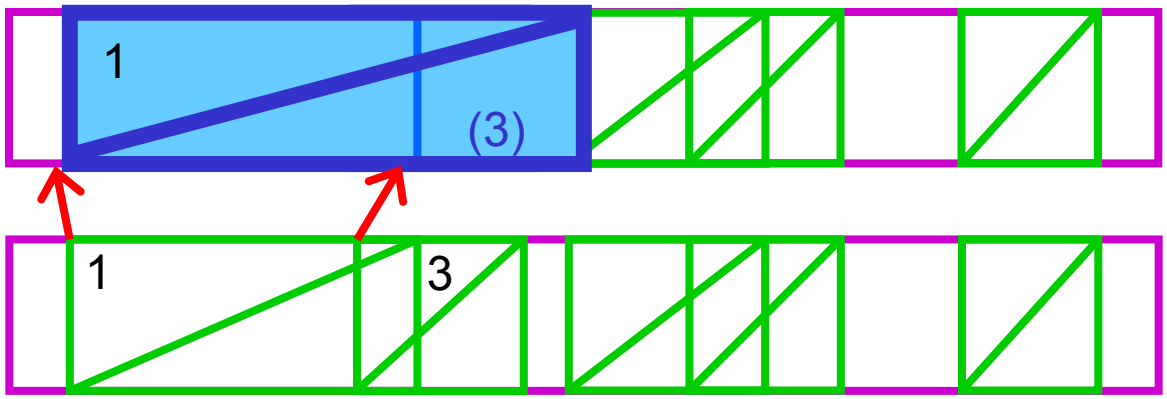
Update x_1', e_1, and w_1:

$$x_1' \leftarrow \frac{e_1 x_1' + e_3\left(x_3' - w_1\right)}{e_1 + e_3} \qquad w_1 \leftarrow w_1 + w_3 \qquad e_1 \leftarrow e_1 + e_3$$

Move cell 1: $\min e_1\left(x_1 - x_1'\right)^2$ ➔ $x_1 = x_1'$

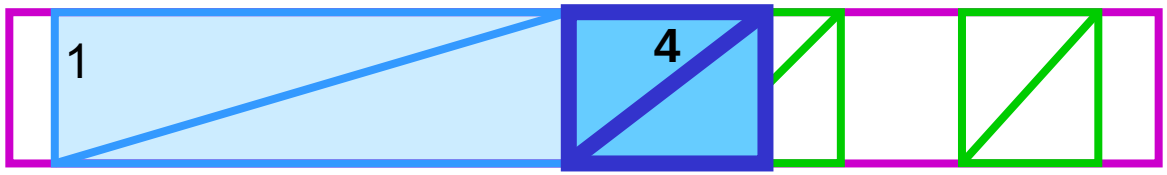
Result:

Prev:



PlaceRow, Cell 4

Cell 4:

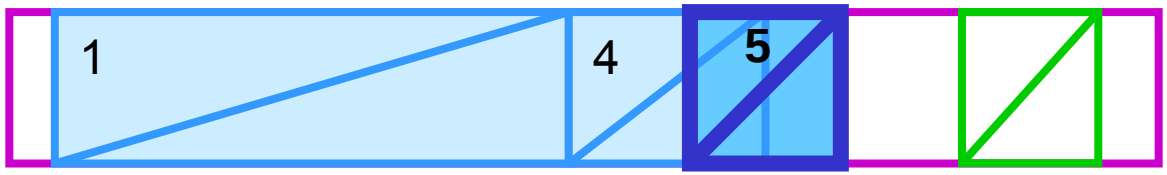


Overlap with previous cell? **no**

➔ **no clustering, no movement**

PlaceRow, Cell 5

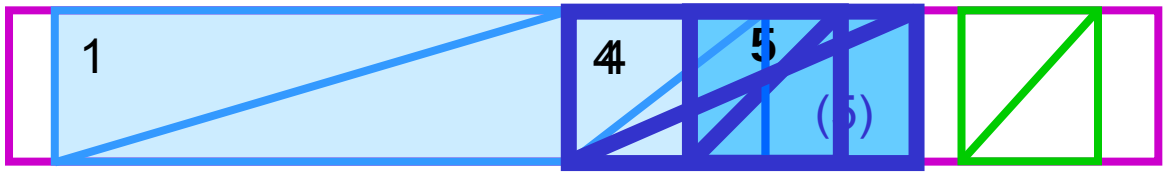
Cell 5:



Overlap with previous cell? **yes** ➔ **cluster** with previous cell 4

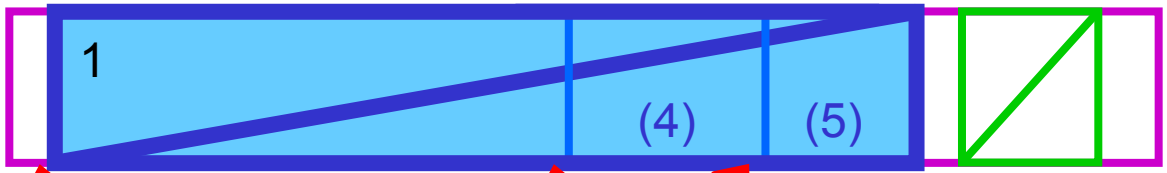
Update x'_4, e_4, and w_4: $x'_4 \leftarrow \dfrac{e_4 x'_4 + e_5\left(x'_5 - w_4\right)}{e_4 + e_5}$ $\quad e_4 \leftarrow e_4 + e_5$ $\quad w_4 \leftarrow w_4 + w_5$

Move cell 4: $\min e_4\left(x_4 - x'_4\right)^2$ ➔ $x_4 = x'_4$

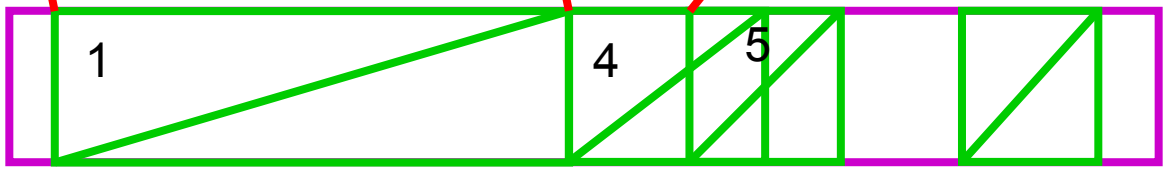


Overlap with previous cell? **yes** ➔ **cluster** with previous cell 1

Result:

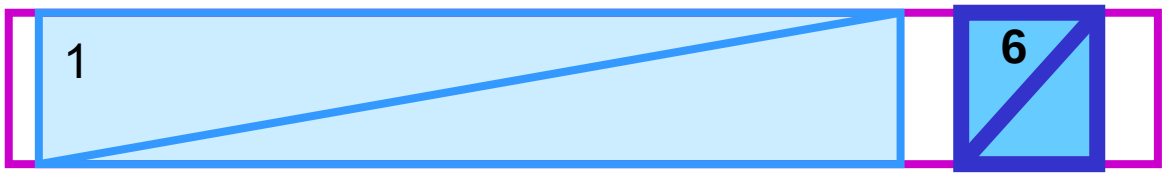


Prev:



PlaceRow, Cell 6

Cell 6:



Overlap with previous cell? **no**

➔ **no clustering, no movement**

Last cell ➔ done, PlaceRow finished

PlaceRow: Summary

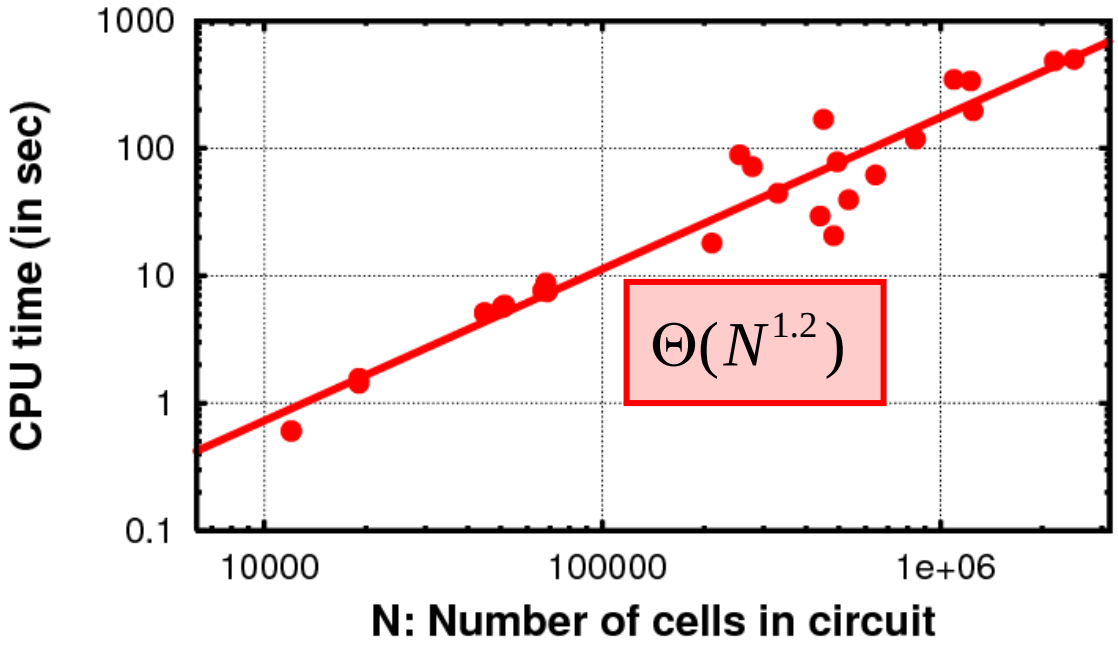
PlaceRow:

- Called several times for legalizing one cell
- Places cells aligned to one row:
  minimize quadratic movement → quadratic program (QP)
- Solves QP by dynamic programming:
  - Process cells from left to right
  - If cell overlaps with previous cell:
    clustering → movement → further checks with left cells
  - Clustering: update width, weight, and global x-pos of cell
    constant execution time
- Linear worst-case complexity: O(N)
  N: number of cells in the row
  At most N-1 clustering operations for N cells

Complexity

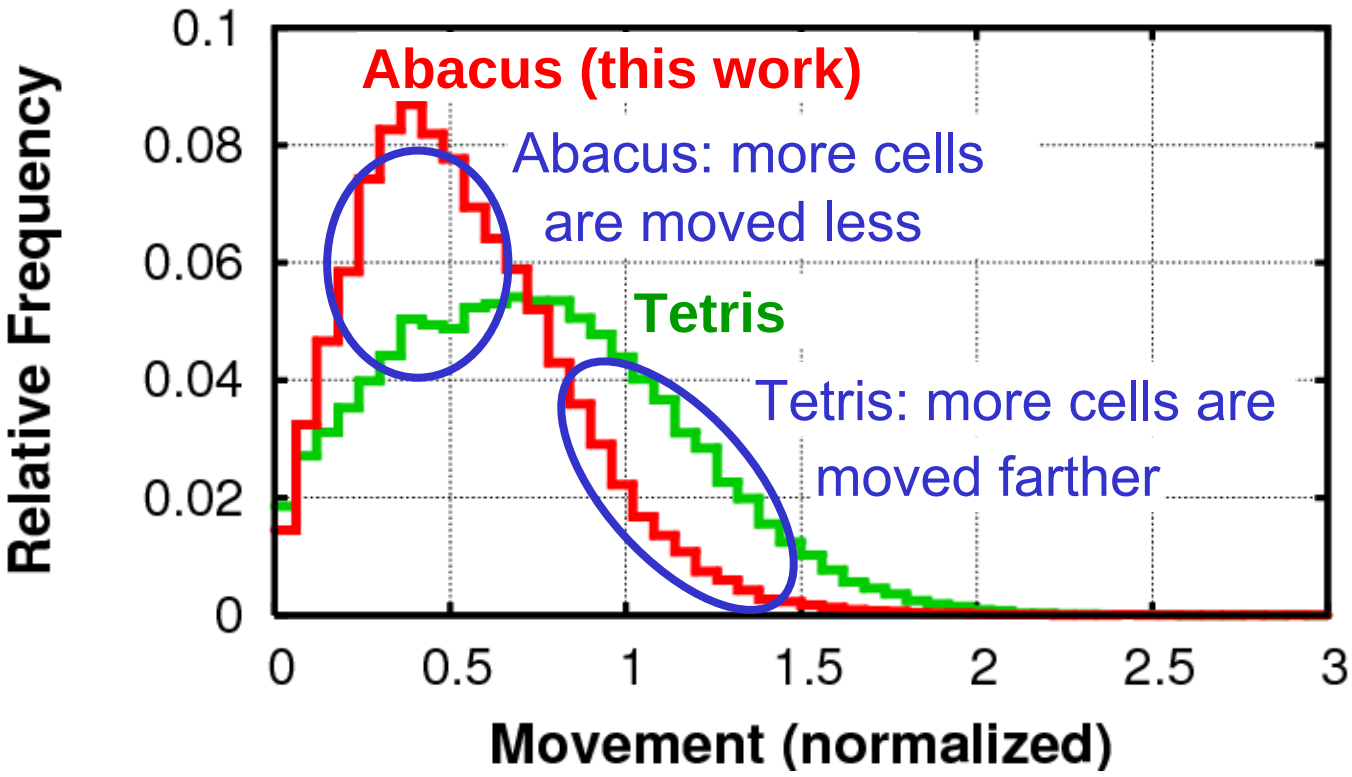
Worst-case for a complete circuit with N cells: $O(N^2)$

Average-case (experimental results):



Movement

Experimental results of one circuit:



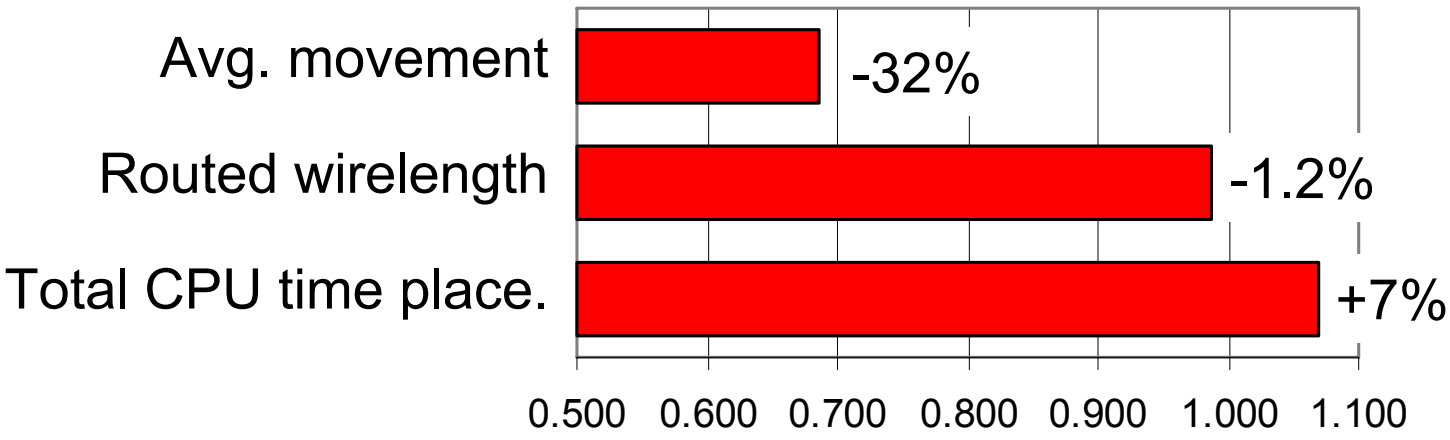
➔ **Lower movement with Abacus**

Results

IBM-Place 2.0 benchmark suite:

- Routability-driven placement
- Global placement: Kraftwerk (with routability optimization)
- Legalization: preserve global placement, minimal movement

Results of Abacus (normalized to Tetris):



Conclusion

Abacus:

- Greedy legalization approach, legalizes one cell at a time, similar to Tetris
- Already legalized cells are moved within the row: PlaceRow
- PlaceRow: minimize total quadratic movement, dynamic programming, linear worst-case complexity
- Results: lower movement than Tetris
  ➔ better results in routability-driven placement

The End

Thanks for the attention!

Questions?