International Symposium on Physical Design

Multi-Electrostatics Based Placement for Non-Integer Multiple-Height Cells

Yu Zhang[1,2], Yuan Pu[1], Fangzhou Liu[1], Peiyu Liao[1], Keren Zhu[1], Kai-Yuan (Kevin) Chao[4], Yibo Lin[2,3*], Bei Yu[1*]

[1]Chinese University of Hong Kong [2]Peking University
[3]Institute of Electronic Design Automation, Peking University
[4]Siemens Digital Industries Software

Outline

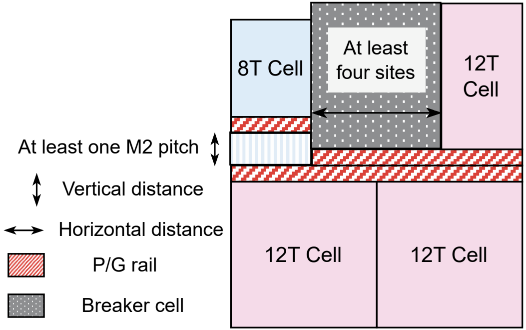
Non-integer Multiple-Height Cell

Standard-cell libraries can be developed with different cell heights, enabling a more flexible optimization of area, timing, and power.

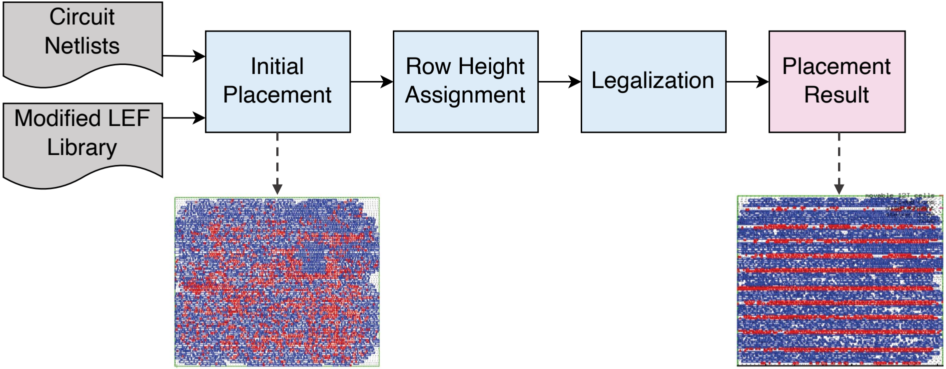
- Large cells provide higher pin accessibility, drive strength, and shorter delay time.
- Small cells have smaller areas, pin capacitance, and power consumption.

Complex Layout Constraints

- C1: at least two cell rows
- C2: even number of rows
- C3: cells placed at sites on rows of the same height
- C4: horizontal spacing
- C5: vertical spacing
- C6: breaker cells insertion

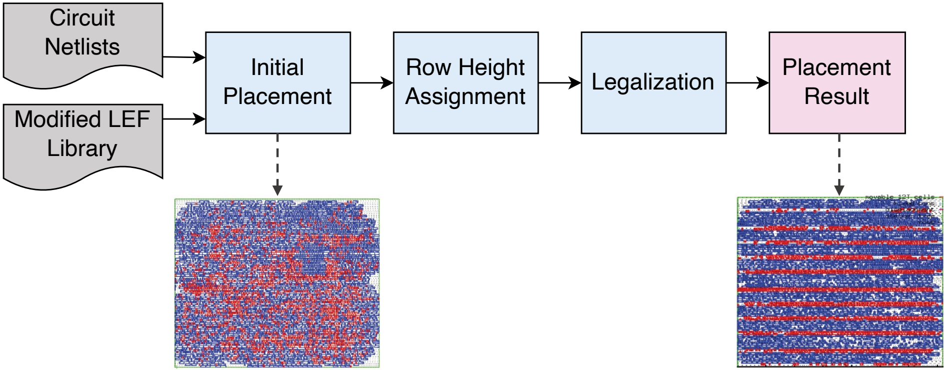


Row-based Placement Flow for NIMH Cells[1]



[1] Zih-Yao Lin and Yao-Wen Chang (2021). "A Row-Based Algorithm for Non-Integer Multiple-Cell-Height Placement". In: *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, pp. 1–6.

Observation



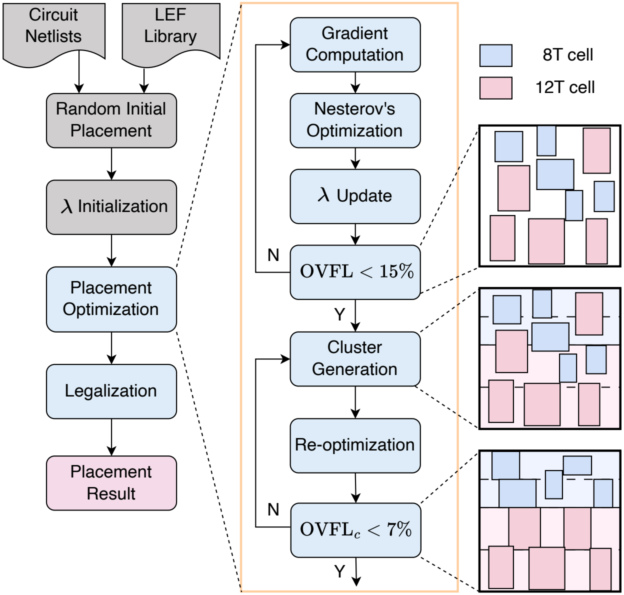
- Traditional flow causes significant disruptions in the initial placement results, resulting in inferior wirelength.

Contribution

Therefore, we propose to

- Adaptively generate regions for each cell type during global placement to identify more desired solutions;
- Introduce a multi-electrostatics-based global placement algorithm to directly solve the global placement problem with NIMH cells.

Overall Flow



The Multi-Electrostatics Based Placement

Considering the density constraint for each cell type c as a distinct electrostatic system, we frame the placement problem with NIMH cells as follows:

$$\min_{x,y} \tilde{W}(x,y) \quad \text{s.t. } \Phi_c(x,y) = 0,\ \forall c \in \mathcal{C}. \tag{1}$$

We leverage the augmented Lagrangian method (ALM) to solve this optimization problem:

$$\min_{x,y} f(x,y) = \tilde{W}(x,y) + \sum_{c \in \mathcal{C}} \lambda_c \left(\Phi_c(x,y) + \frac{1}{2}\mu\theta_\lambda \Phi_c(x,y)^2\right), \tag{2}$$

where λ_c represents the density multiplier for each cell type.

Benefits of Augmented Lagrangian Method

- The constrained optimization problem is transformed into an unconstrained optimization problem;
- The ALM formulation can be interpreted as a combination of the multiplier method and the quadratic penalty method.

Gradient Computation and Preconditioning

The x-directed gradient of our ALM objective function can be derived as follows:

$$\frac{\partial f(x,y)}{\partial x_i} = \frac{\partial \tilde{W}(x,y)}{\partial x_i} + \lambda_c\left(\frac{\partial \Phi_c(x,y)}{\partial x_i} + \mu\theta_\lambda \Phi_c(x,y)\frac{\partial \Phi_c(x,y)}{\partial x_i}\right), \forall i \in \mathcal{V}_c. \tag{3}$$

Then, the preconditioned[2] gradient would be input into Nesterov's optimizer[3] for a gradient descent step.

[2]Myung-Chul Kim and Igor L Markov (2012). "ComPLx: A competitive primal-dual lagrange optimization for global placement". In: *Proceedings of the 49th Annual Design Automation Conference (DAC)*, pp. 747–752.

[3]Jingwei Lu et al. (2015). "ePlace: Electrostatics-based placement using fast fourier transform and Nesterov's method". In: *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 20.2, pp. 1–34.

Density Multipliers Update

Given that the dual function, $Z(\lambda) = \max f(x, y)|_{\lambda}$, associated with Eq. 2 is not smooth but piecewise linear, we utilize the subgradient method to update λ as,

$$\lambda^{k+1} \leftarrow \min(\lambda_{\max}, \max(0, \lambda^k + \alpha^k g_{\text{sub}}(\lambda))), \tag{4}$$

where $g_{\text{sub}}(\lambda) = (\ldots, \Phi_c(x, y) + \frac{1}{2}\mu\theta_{\lambda}\Phi_c(x, y)^2, \ldots)$. However, the convergence of the traditional subgradient method highly depends on α^k.

Surrogate Subgradient Method

The main idea of the surrogate subgradient[4] method is to obtain α^k such that distances between Lagrangian multipliers λ^k at consecutive iterations decrease, i.e.,

$$\|\lambda^{k+1} - \lambda^k\| = \eta^k \|\lambda^k - \lambda^{k-1}\|, \tag{5}$$

where $0 < \eta^k < 1$. Eq. 4 and Eq. 5 imply

$$\alpha^k = \eta^k \frac{\alpha^{k-1} \|g_{\text{sub}}(f^{k-1})\|}{\|g_{\text{sub}}(f^k)\|}. \tag{6}$$

[4]Mikhail A Bragin et al. (2015). "Convergence of the surrogate Lagrangian relaxation method". In: *Journal of Optimization Theory and applications* 164, pp. 173–201.

Considering NIMH constraints, we prioritize clustering cells of the same type together. To achieve this, we introduce pseudo-nets for cells with the same height. The score function $c(i,j)$ in the modified BestChoice clustering algorithm is defined as follows:

$$c(i,j) = \sum_{e \in E_{i,j}} \frac{\omega_e}{a_i + a_j}, \tag{7}$$

where ω_e is a corresponding edge weight defined as:

$$\omega_e = \begin{cases} e^{\frac{1}{|x_i - x_j| + |y_i - y_j|}}, & h_i = h_j \text{ and e is a real net}, \\ 1, & h_i = h_j \text{ and e is a pseudo-net}, \\ 0, & h_i \neq h_j. \end{cases} \tag{8}$$

Dataset

We conducted experiments using eight design blocks (`sha3`, `aes_core`, `des`, `fpu`, `des3`, `mor1kx`, `jpeg`, `aes_128`) obtained from the OpenCores website.

Table: Statistics of the OpenCores benchmarks

Design	#Cells	#Nets	Util (%)	Clock (ps)
sha3	1337	1397	69.05	100
aes_core	4733	4808	69.84	400
des	18274	18372	67.11	250
fpu	30495	31225	67.65	270
des3	58017	58116	67.05	250
mor1kx	61220	58952	67.32	200
jpeg	210968	233898	68.49	300
aes_128	250672	225888	57.29	300

Experimental Result

Table: WNS (ns), TNS (ns), HPWL (10^5um) and CPU Runtime (s) with State-of-the-art Row-based Placers.

test case	Cells			ICCAD'21-imp				Ours			
	8T	12T	Total	WNS (ns)	TNS (ns)	HPWL (10^5um)	Runtime (s)	WNS (ns)	TNS (ns)	HPWL (10^5um)	Runtime (s)
sha3	662	675	1337	-0.09	-1.89	3.96	45.3	-0.09	-1.85	3.25	23.07
aes_core	2511	2222	4733	-0.15	-22.16	4.38	78.36	-0.14	-19.14	3.76	23.76
des	8853	9421	18274	0.11	0	14.50	218.90	0.11	0	12.21	27.58
fpu	15266	15229	30495	-0.22	-4.37	25.14	315.88	-0.17	-3.28	23.63	31.26
des3	29683	28334	58017	-0.05	-0.20	46.78	564.51	-0.04	-0.11	37.85	33.58
mor1kx	30168	29873	60041	-0.13	-4.10	61.22	808.22	-0.13	-4.49	63.20	32.86
jpeg	107866	103102	210968	-0.48	-128.20	152.66	2528.35	-0.36	-66.10	149.67	59.15
aes_128	123825	126847	250672	-0.32	-61.97	221.90	2569.14	-0.25	-54.33	178.30	72.03
average ratio				1.22	1.49	1.12	23.50	**1.00**	**1.00**	**1.00**	**1.00**

Conclusion

- On average, our method achieves a 12% reduction in HPWL while exhibiting a remarkable 23.5$\times$ faster runtime.
- In large cases involving over 200,000 standard cells, our method shows up to 42.85$\times$ speedup while delivering better placement solution quality.
- Our method improves 22% and 49% in WNS and TNS, respectively.

THANK YOU!