ASIA SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE
ASP-DAC 2025
30th Asia and South Pacific Design Automation Conference

FTAFP: A Feedthrough-Aware Floorplanner for Hierarchical Design of Large-Scale SoCs

Zirui Li[1,*], **Kanglin Tian**[1,*], Jianwang Zhai[1,†], Zixuan Li[1], Shixiong Kai[2], Siyuan Xu[2], Bei Yu[3], Kang Zhao[1]

[1]Beijing University of Posts and Telecommunications
[2]Huawai Noah's Ark Lab
[3]The Chinese University of Hong Kong

Jan. 21, 2025

Outline

- Introduction
- Preliminaries
- Framework
- Evaluation
- Conclusion

Introduction——Background

- As the scale and complexity of System-on-Chips (SoCs) continue to grow, **hierarchical** and **modular** concepts pushing the **floorplanning** challenge down to the **sub-chip level**.

- Hierarchical breaks complex systems down into multiple levels of **subsystems**, modular design to **package and reuse** different functionalities at each level.

- These integrated, bottom-up design methods significantly speed up the front-end chip design process.

Introduction——Challenge

- However, these methods also present new optimization challenge in **floorplanning**, named Feedthrough.

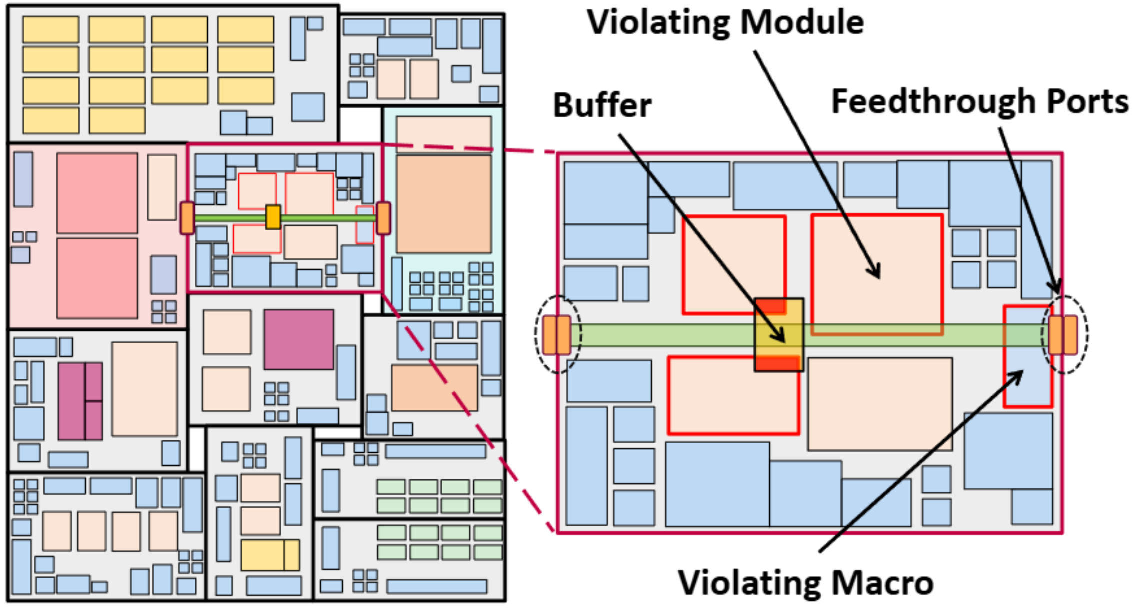


Fig.1. The violations caused by feedthrough insertion in hierarchical floorplanning of the large-scale SoC.

- Feedthrough is a through-module connection, yet it would require additional buffers and ports inside the module for data transmission.

Introduction——Related works

- Analytical-based Methods[1]
  - Generally adopt a two-stage framework of global distribution and legalization.
- Heuristic Methods[2]
  - Rely on topological representations and employ heuristic algorithms to optimize floorplans.
- Learning-based Methods[3]
  - learning an optimized and generalized mapping between circuit connectivity to produce a chip floorplan.

[1]F. Huang, et al. "Handling orientation and aspect ratio of modules in electrostatics-based large scale fixed-outline floorplanning," In: *Proc. ICCAD*, 2023.
[2]Y.-C. Chang, et al. "B*-Trees: a new representation for non-slicing floorplans," In: *Proc. DAC*, 2000.
[3]Y. Liu, et al. "GraphPlanner: Floorplanning with graph neural network," In: *ACM TODAES*, 2022.

Introduction——Limitations

In order to minimize feedthrough, the above methods has the following limitations:

- **Overlaps in the global distribution stage** of analytical methods complicate feedthrough handling, which relies on neighbor information.
- Learning-based methods, **constrained in representation and generalization**, are also unsuitable for feedthrough optimization.

Considering both efficiency and generality, **heuristic-based methods** are promising approaches to address the feedthrough challenge.

Outline

- Introduction
- Preliminaries
- Framework
- Evaluation
- Conclusion

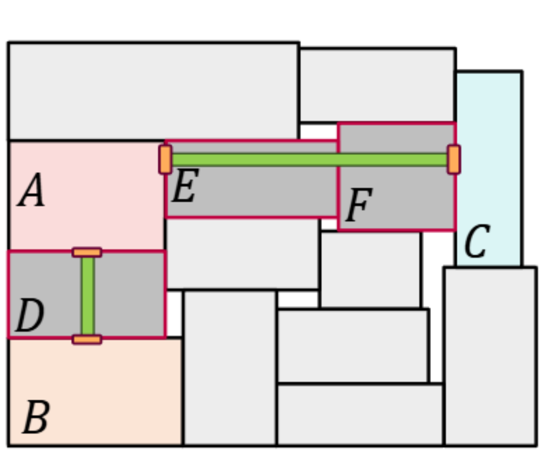
Preliminaries

- **Fixed-outline Floorplanning**
- Let $B = \{b_i | 1 \leq i \leq n\}$ be a set of rectangle modules, each module b_i has width w_i and height h_i.
- The connections among modules are described in netlist $N = \{N_i | i \leq i \leq m\}$, where each N_i specifies a set of modules requiring connectivity.
- The fixed-outline floorplanning aims to place all modules without overlapping in a rectangular outline R, with width W_0 and height H_0.
- Given the total modules' area A and a maximum white space ratio σ, the width W_0 and the height H_0 are calculated as:

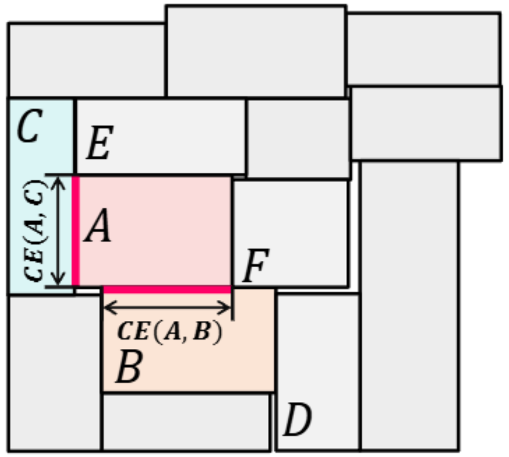
$$W_0 = \sqrt{(1+\sigma)A\lambda}, H_0 = \sqrt{(1+\sigma)A/\lambda}$$

Preliminaries

- **Feedthrough Problem**
- Feedthrough
  - Feedthrough wirelength FTH_{wl} and number of feedthroughed modules FTH_{num}



(a)



(b)

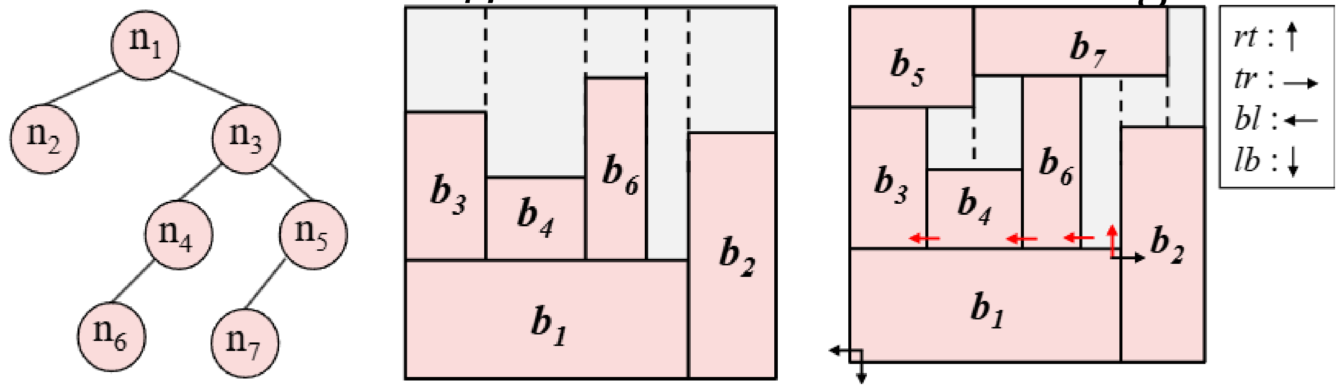
Fig.2. Feedthrough example with two nets, $N_1 = \{A, B\}$ and $N_2 = \{A, C\}$.

- Common Edge
  - If A, B adjacent vertically,

$$ce_{len}(A,B) = min(A.x_{tr}, B.x_{tr}) - max(A.x_{bl}, B.x_{bl})$$

Preliminaries

- **CB-Tree Representation**
- CB-Tree
  - A CB-Tree is a B*-Tree integrated with corner stitching.



(a) (b) (c)

Fig.3. (a) A CB-Tree example. (b) Subtree packing. (c) Neighbor finding.

- Corner Stitching
  - A classical data structure for representing non-overlapping rectangular modules in the 2D plane (called tile plane).
- Neighbor Finding
  - Corner stitching provides many efficient operations to support handling geometric constraints such as neighbor finding.

Preliminaries

- **Slack Computation**

The slack of a module refers to the range within which it can move without overlapping or pushing other modules.

- The x and y coordinates of modules are computed separately.

- In each dimension, the floorplan is constrained by one or more “critical paths” in corresponding constraint graphs.

- Any change in the location of a module on the critical path will produce overlaps or increase the span of the floorplan.

- Module’s horizontal slack (x-slack)is calculated as

$$slack(v_{h_i}) = R(v_{h_i}) - L(v_{h_i})$$

Outline

- Introduction
- Preliminaries
- Framework
- Evaluation
- Conclusion

Framework

- **Overview**
- **Feedthrough Estimation Model**
- **Slack Computation by CS**
- **Two-phase SA Framework**
- **Cost Evaluation**

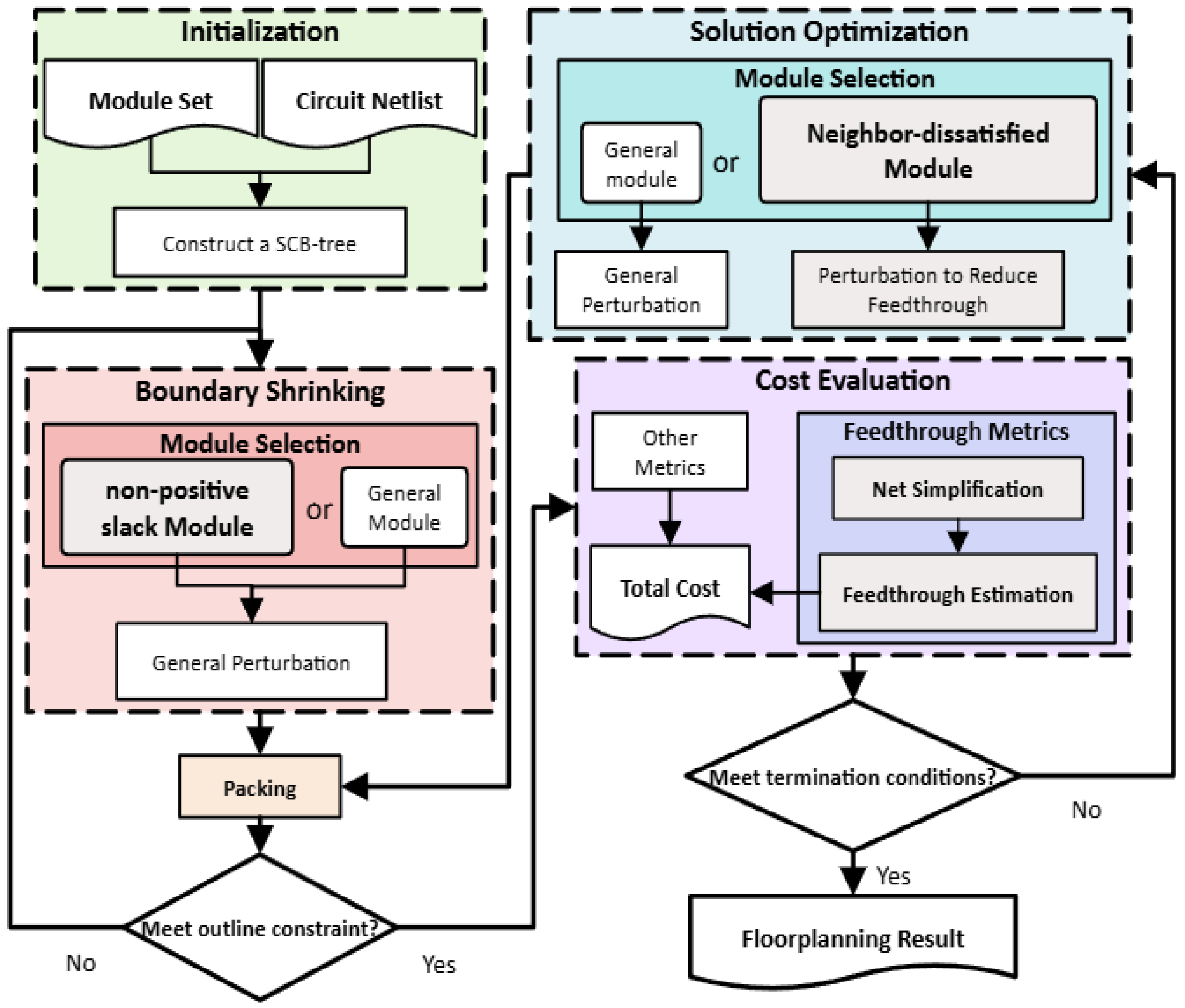


Fig.4. Flowchart of FTAFP

Framework

- **Feedthrough Estimation Model**
  - **Net Simplification**

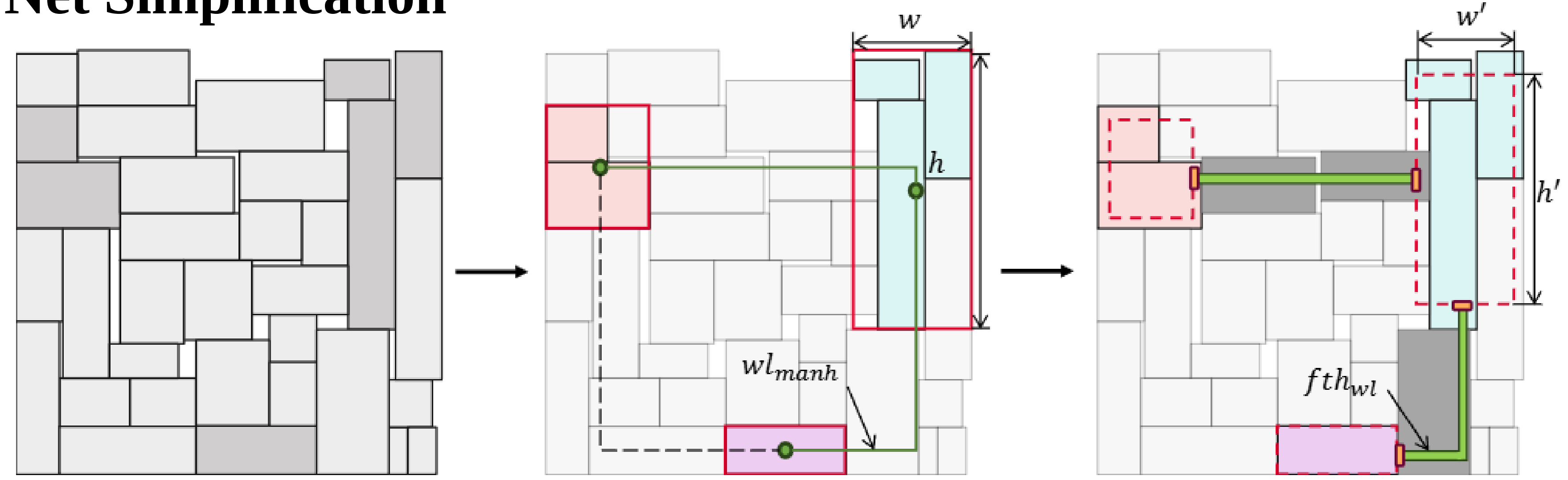


Fig.5. An example of net simplification. A net with 6 modules which are clustered into 3 sub-nets

- Transform nets into undirected graphs
- Merge adjacent modules and computing common edge length
- Found shortest feedthrough connections by using MST

Framework

- **Feedthrough Estimation**
  - fth_{wl}: For net N_i, its $fth_{wl}(N_i)$ can be estimated as the sum of the wirelength of each feedthrough edge within the MST.

$$fth_{wl}(N_i) = \sum_{e(A,B)\in E_i} fth_{wl}(A,B) \times \omega(A,B)$$

$$fth_{wl}(A,B) = [wl_{manh}(A,B) - wl'(A) - wl'(B)] \times \omega(A,B)$$

$$wl'(A) = \frac{w(A) + h(A)}{2} \times \sqrt{\frac{area'(A)}{area(A)}}$$

  - fth_{num}: To estimate fth_{num}, we propose a greedy detection algorithm based on neighbor searching.

Framework

- **Slack Computation by Corner Stitching**
  - **Slack Computation**

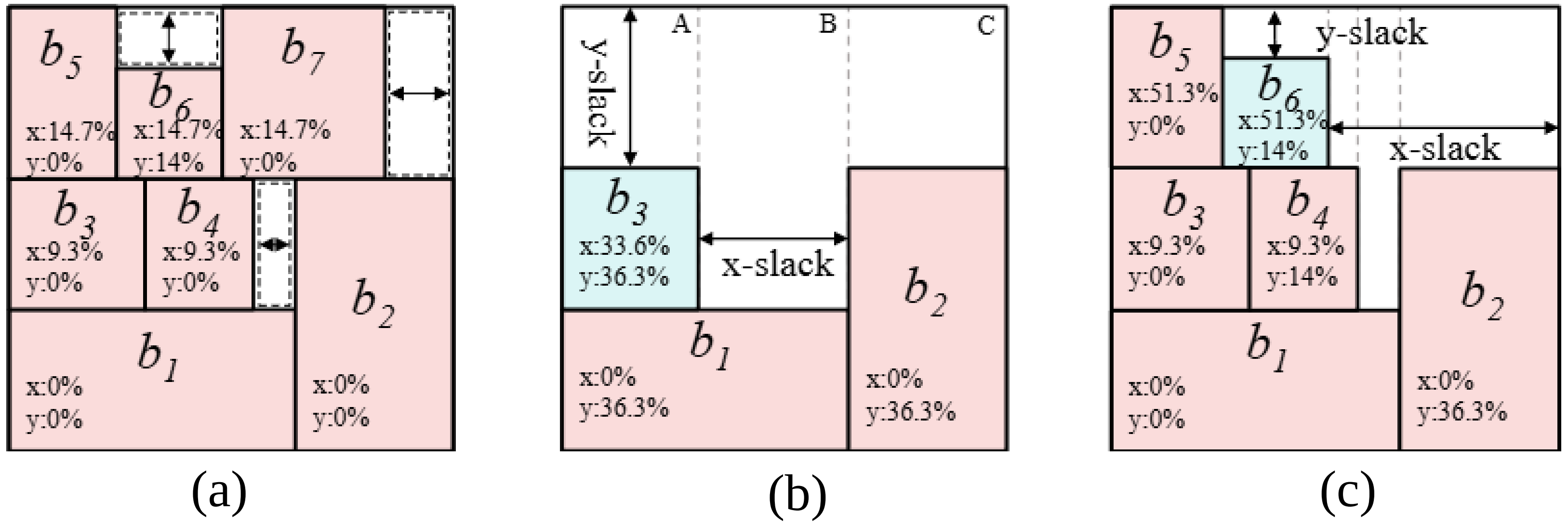


Fig.6. (a) An example tile plane contains seven packed modules. (b) Slacks of b_3 are initialized its slacks based on the associated tiles A and B. (c) When packing module b_6, we update module b_5's x-slack and module b_4's y-slack as b_6's.

Framework

- **Slack Computation by Corner Stitching**
  - **The SCB-Tree Packing Flow**

  - Determine the positions of modules
  - Initializing the slack of modules
  - Update the slack of modules by neighbor finding operation

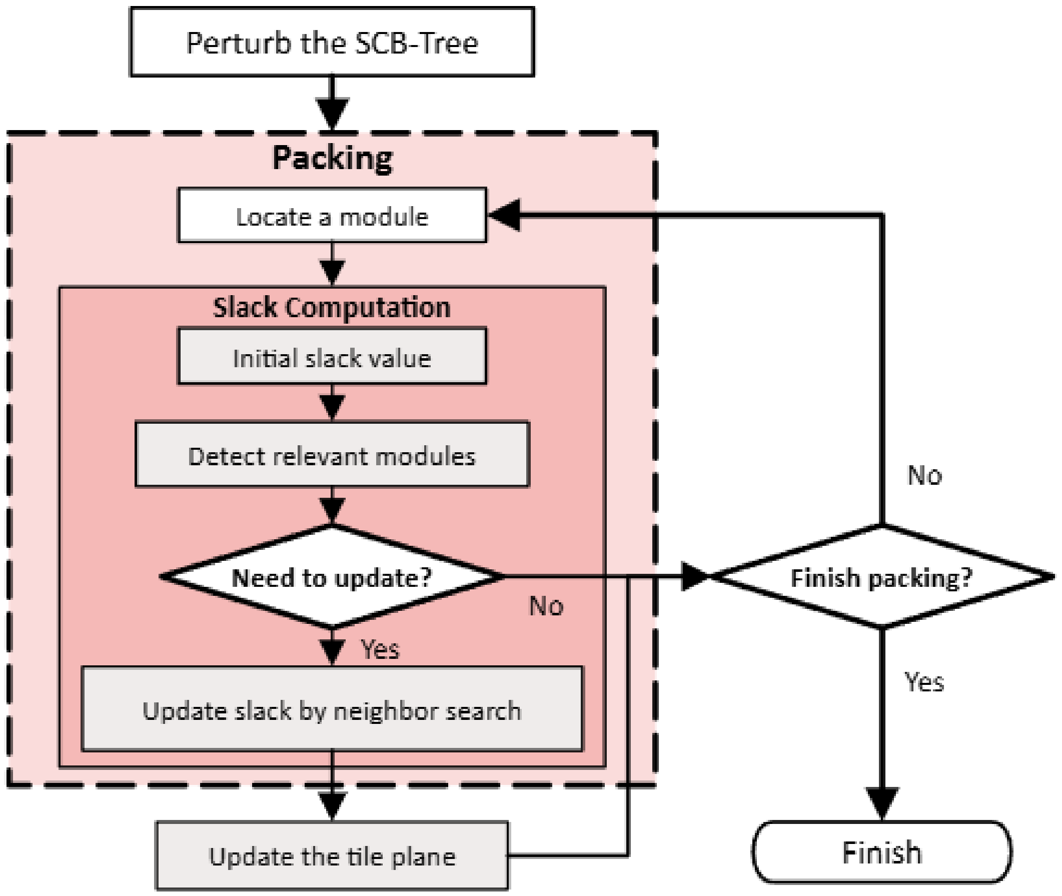


Fig.7. The SCB-Tree packing flow

Framework

- **Two-phase SA Framework**
  - **Boundary Shrinking Phase**

    The B*-Tree is commonly used for three main operations:

    - Op1: Rotate a module.
    - Op2: Relocate a module.
    - Op3: Swap two modules.
  - **Solution Optimization Phase**
    - Op4: Change the module's neighbors. Select a module with minimal neighbor satisfaction and randomly swap it to its neighborhood-demanding child nodes.

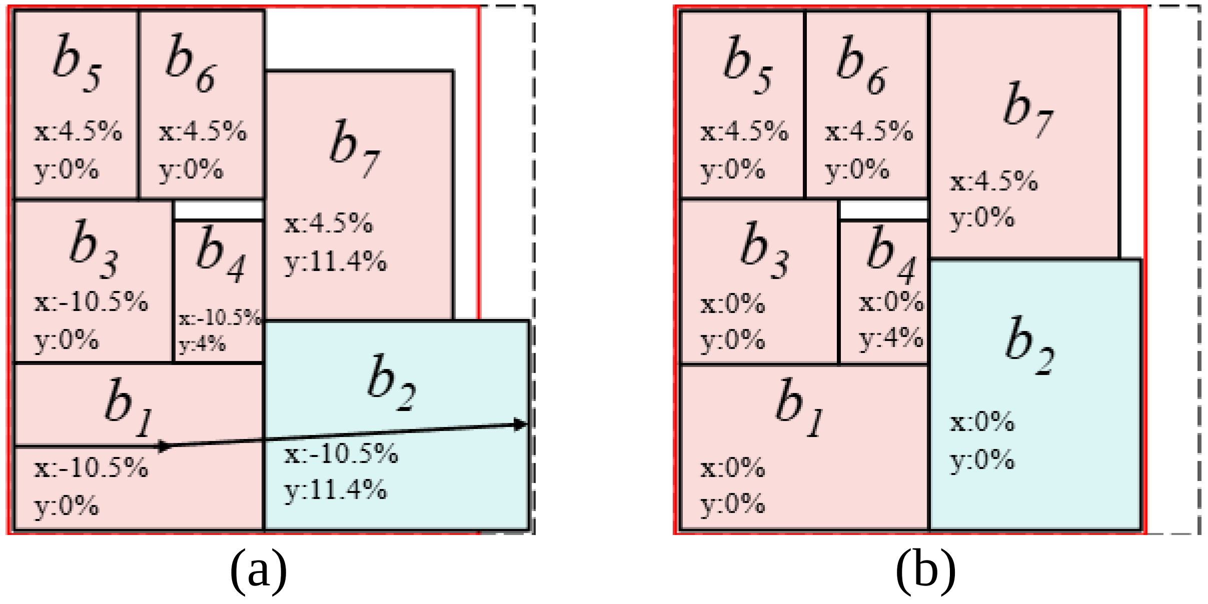


Fig.8. (a) A floorplan with 7 modules, the black arrow represents the critical path on x-dimension, and the x-slack on this path is non-positive. (b) After rotating module b_2 on the critical path, the x-span of the floorplan is reduced.

Framework

- **Cost Evaluation**
- The cost of the SA is calculated by summing weighted metrics. However, these metrics have a large range and cannot be simply weighted sum. Therefore, we develop a method to calculate the cost of the SA:
  - Expectations of each metric: $\overline{Metrics}$
  - Normalizing: $\frac{\overline{Metrics}}{Metrics}$
  - Weighted sum: $\alpha A + \beta B + \gamma C + \delta D$
- Finally, we can obtain the total cost, defined as:

$$cost = \alpha \frac{Area_{total}}{Area} + \beta \frac{\overline{HPWL}}{HPWL} + \gamma \frac{\overline{CE_{len}}}{CE_{len}} + \delta \left(\frac{\overline{FTH_{num}}}{FTH_{num}} + \frac{\overline{FTH_{wl}}}{FTH_{wl}} \right)$$

Outline

Evaluation

- **Experiments Settings**

- The FTAFP framework is compared with three competitive heuristic-based methods based on the topological representation: Corblivar[4], SP-FOFP[5], and CB-Tree[6].

- The test cases are derived from the GSRC[7] and MCNC[8] benchmarks.

---

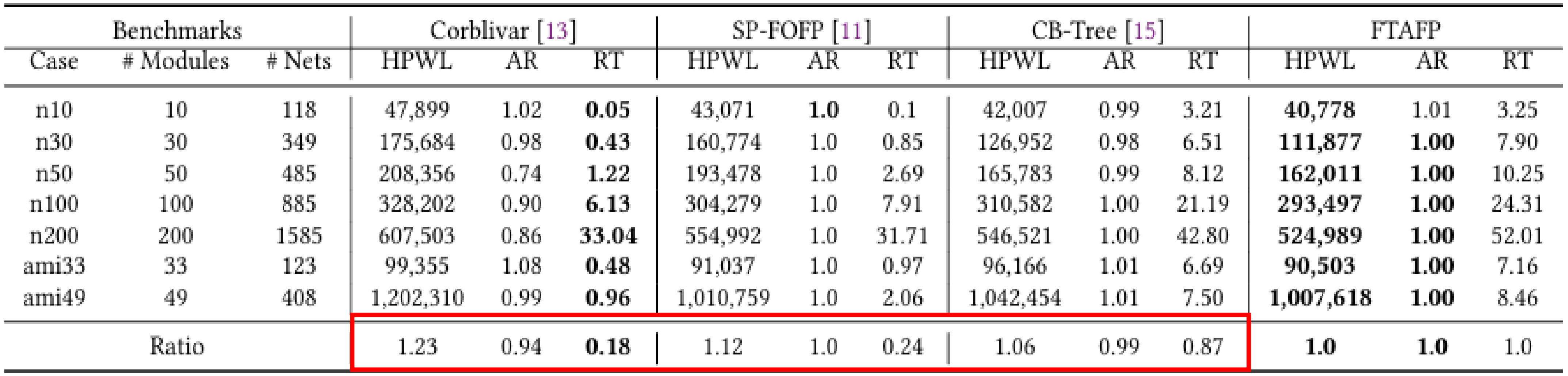
[4]J. Knechtel, et al. “Structural planning of 3D-IC interconnects by block alignment,” In: *Proc. ASPDAC,* 2014.
[5]Q. Xu, et al. “Combining the ant system algorithm and simulated annealing for 3D/2D fixed-outline floorplanning,” Elsevier Applied Soft Computing, 2016.
[6]H.-F. Tsao, et al. “A corner stitching compliant B*-tree representation and its applications to analog placement,” In: *Proc. ICCAD* 2011.
[7]W. Dai, L. Wu, and S. Zhang. (2000) GSRC benchmarks. [Online]. Available: http://vlsicad.eecs.umich.edu/BK/GSRCbench/
[8]M. C. of North Carolina (MCNC). (2000) MCNC benchmarks.[Online]. Available: http://vlsicad.eecs.umich.edu/BK/MCNCbench/

Evaluation

- **Results without Feedthrough Optimization**

- Average wirelength reductions of 23%, 12%, and 6% over Corblivar, SP-FOFP, and CB-Tree.
- Increase in the runtime of around 13% compared to CB-Tree.

Table 1: Comparison of baselines with FTAFP, without feedthrough optimization.



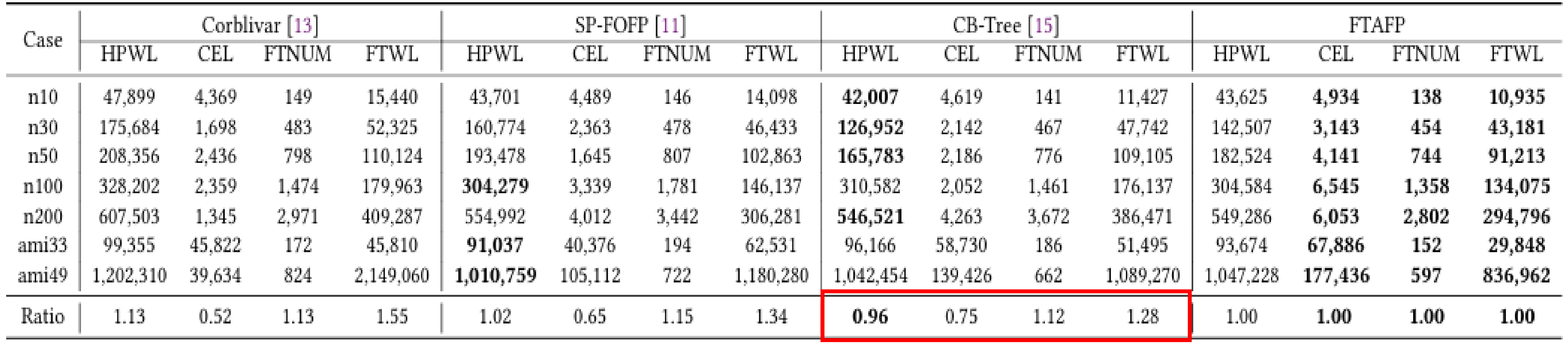
Benchmarks			Corblivar [13]			SP-FOFP [11]			CB-Tree [15]			FTAFP		
Case	# Modules	# Nets	HPWL	AR	RT	HPWL	AR	RT	HPWL	AR	RT	HPWL	AR	RT
n10	10	118	47,899	1.02	**0.05**	43,071	**1.0**	0.1	42,007	0.99	3.21	**40,778**	1.01	3.25
n30	30	349	175,684	0.98	**0.43**	160,774	1.0	0.85	126,952	0.98	6.51	**111,877**	**1.00**	7.90
n50	50	485	208,356	0.74	**1.22**	193,478	1.0	2.69	165,783	0.99	8.12	**162,011**	**1.00**	10.25
n100	100	885	328,202	0.90	**6.13**	304,279	1.0	7.91	310,582	1.00	21.19	**293,497**	**1.00**	24.31
n200	200	1585	607,503	0.86	**33.04**	554,992	1.0	31.71	546,521	1.00	42.80	**524,989**	**1.00**	52.01
ami33	33	123	99,355	1.08	**0.48**	91,037	1.0	0.97	96,166	1.01	6.69	**90,503**	**1.00**	7.16
ami49	49	408	1,202,310	0.99	**0.96**	1,010,759	1.0	2.06	1,042,454	1.01	7.50	**1,007,618**	**1.00**	8.46
Ratio			1.23	0.94	**0.18**	1.12	1.0	0.24	1.06	0.99	0.87	**1.0**	**1.0**	1.0

Evaluation

- **Results with Feedthrough Optimization**

- Compared to the CB-Tree, the FTNUM and FTWL metrics are reduced by 12% and 28%, and CEL is also improved by 25%.

Table 2: Comparison of HPWL, common edge length (CEL), feedthrough number (FTNUM) and feedthrough wirelength (FTWL).



Case	Corblivar [13]				SP-FOFP [11]				CB-Tree [15]				FTAFP			
	HPWL	CEL	FTNUM	FTWL	HPWL	CEL	FTNUM	FTWL	HPWL	CEL	FTNUM	FTWL	HPWL	CEL	FTNUM	FTWL
n10	47,899	4,369	149	15,440	43,701	4,489	146	14,098	**42,007**	4,619	141	11,427	43,625	**4,934**	**138**	**10,935**
n30	175,684	1,698	483	52,325	160,774	2,363	478	46,433	**126,952**	2,142	467	47,742	142,507	**3,143**	**454**	**43,181**
n50	208,356	2,436	798	110,124	193,478	1,645	807	102,863	**165,783**	2,186	776	109,105	182,524	**4,141**	**744**	**91,213**
n100	328,202	2,359	1,474	179,963	**304,279**	3,339	1,781	146,137	310,582	2,052	1,461	176,137	304,584	**6,545**	**1,358**	**134,075**
n200	607,503	1,345	2,971	409,287	554,992	4,012	3,442	306,281	**546,521**	4,263	3,672	386,471	549,286	**6,053**	**2,802**	**294,796**
ami33	99,355	45,822	172	45,810	**91,037**	40,376	194	62,531	96,166	58,730	186	51,495	93,674	**67,886**	**152**	**29,848**
ami49	1,202,310	39,634	824	2,149,060	**1,010,759**	105,112	722	1,180,280	1,042,454	139,426	662	1,089,270	1,047,228	**177,436**	**597**	**836,962**
Ratio	1.13	0.52	1.13	1.55	1.02	0.65	1.15	1.34	**0.96**	0.75	1.12	1.28	1.00	**1.00**	**1.00**	**1.00**

Outline

- Introduction
- Preliminaries
- Framework
- Evaluation
- Conclusion

Conclusion

- **We propose a feedthrough-aware floorplanner named FTAFP** to solve the feedthrough challenge faced by the hierarchical design of large-scale SoCs.
- To the best of our knowledge, we are **the first to model and optimize the feedthrough problem in floorplanning**.
- We **introduce SCB-Tree to better satisfy the fixed-outline constraint** and optimization objectives and **propose a two-phase SA framework** with targeted perturbation operations.

THANK YOU!