Security-aware Physical Design against Trojan Insertion, Frontside Probing, and Fault Injection Attacks

ISPD'23

Jhih-Wei Hsu, Kuan-Cheng Chen, Yan-Syuan Chen, Yu-Hsiang Lo, Yao-Wen Chang

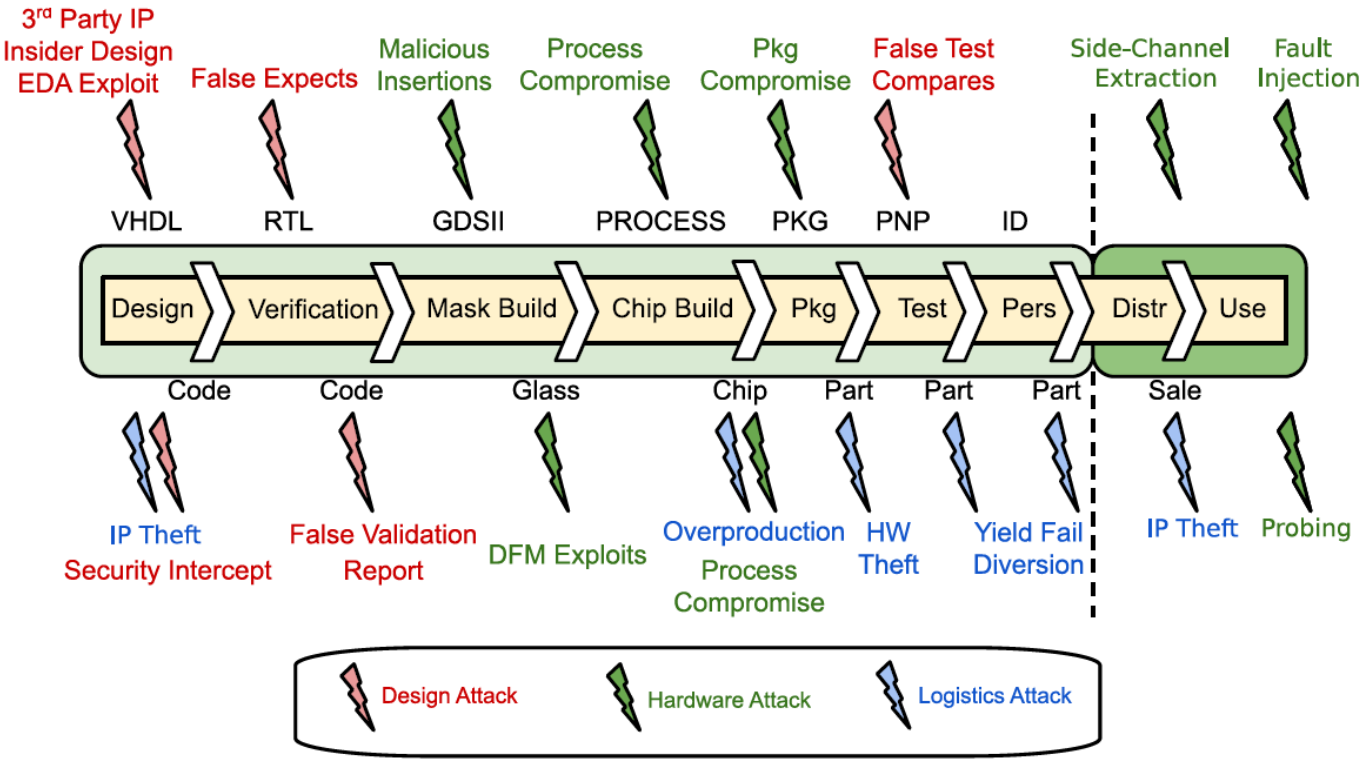
The Electronic Design Automation Laboratory
Graduate Institute of Electronics Engineering
National Taiwan University
Taipei 106, Taiwan

Outline

- Introduction
- Preliminaries
- Proposed Framework
- Experimental Results
- Conclusions

Introduction

- IC companies start outsourcing the design and production of the chips to lower the manufacturing cost
  - The fabrication process requires many stages from upstream to downstream
  - Attackers could have plenty of opportunities to modify ICs maliciously



Different security threats during the life-cycle of ICs [Knechtel et al., ICCAD'21]

Hardware Security

- Hardware security has become a critical issue due to the globalization of the semiconductor supply chain
  - Unlike software, ICs are not patchable after manufacturing
  - Efforts made at higher levels may be undermined in later PPA (power, performance, and area) optimization

- **Secure-by-design** is a paradigm of computer-aided design (CAD)
  - Consider confidentiality, integrity, and availability throughout the CAD flow
  - Protect ICs proactively
  - Ensure the effects of security methods remain consistent in the following stages

Contributions

- Propose an effective defense framework
  - A large-scale shielding method can effectively protect all the cell and net assets from frontside probing and fault injection attacks
  - A cell movement-based method can effectively protect layouts from Trojan insertion attacks
- Our proposed framework also considers comprehensive design cost (PPA)
- Experiment results show that our proposed framework can significantly reduce the attack metrics
  - Achieve the best score among the participating teams at the 2022 ACM ISPD Contest

Outline

- Introduction
- Preliminaries
- Proposed Framework
- Experimental Results
- Conclusions

Trojan Insertion Attack (1/2)

- A malicious hardware modification leads to incorrect behavior during operation
  - Targeting at the system level, behavior level, gate level, or physical level
  - Intending to leak sensitive information or destroy the functionality of the circuit

- Most Trojans comprise a trigger and a payload
  - Trigger: to monitor the circuit signals and activates the payload when an expected event occurs
  - Payload: to perform the actual attack when receiving the signal from the trigger

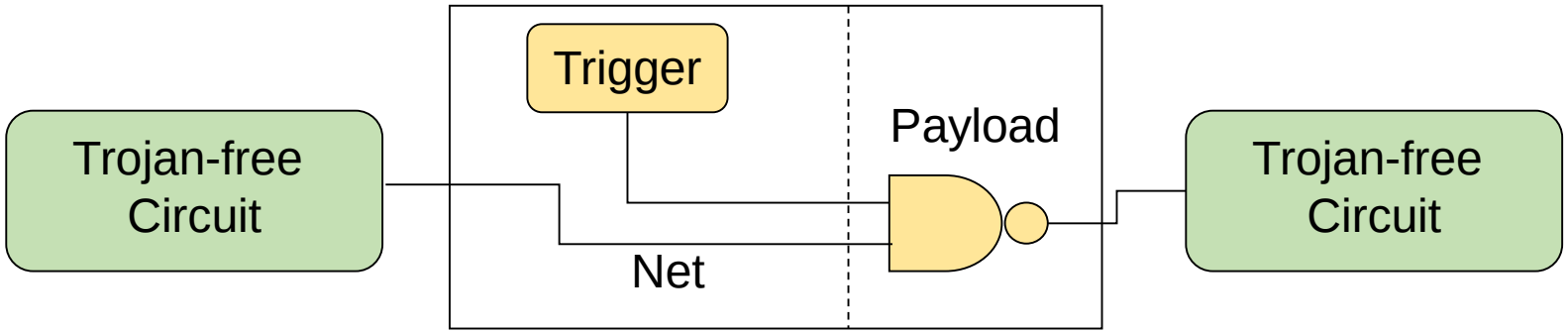


Illustration of hardware Trojan

Trojan Insertion Attack (2/2)

- A successful Trojan attack requires the three conditions
  - Trojan placement: a spatial space to place the additional Trojan components
  - Trojan integration: a connection between the Trojan payload and security components
  - Intra-Trojan routing: a connection between the trigger and payload of the Trojan

- This work focuses on post-design time gate-level addictive Trojan attacks

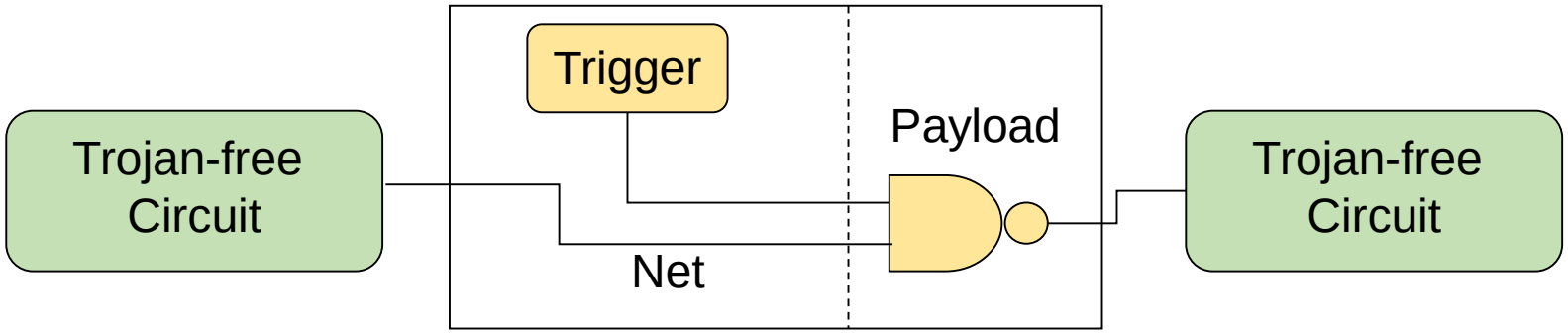


Illustration of hardware Trojan

Frontside Probing Attacks

- An invasive physical attack that enables the probe to expose the cells or nets to extract sensitive information
  - Using contact-based micro-probing, electromagnetic field probing, or electro-optical device probing to achieve an attack
  - Obtain chip information, acquire device configuration, or destroy system functionality
- This work focuses on the frontside contact-based probing attack

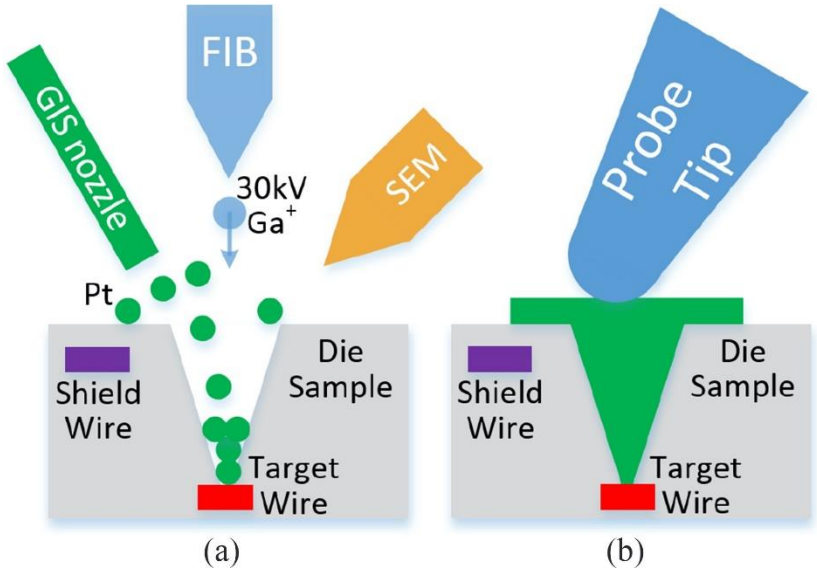


Illustration of a focused ion beam (FIB)
[Wang et al., TCAD'20]

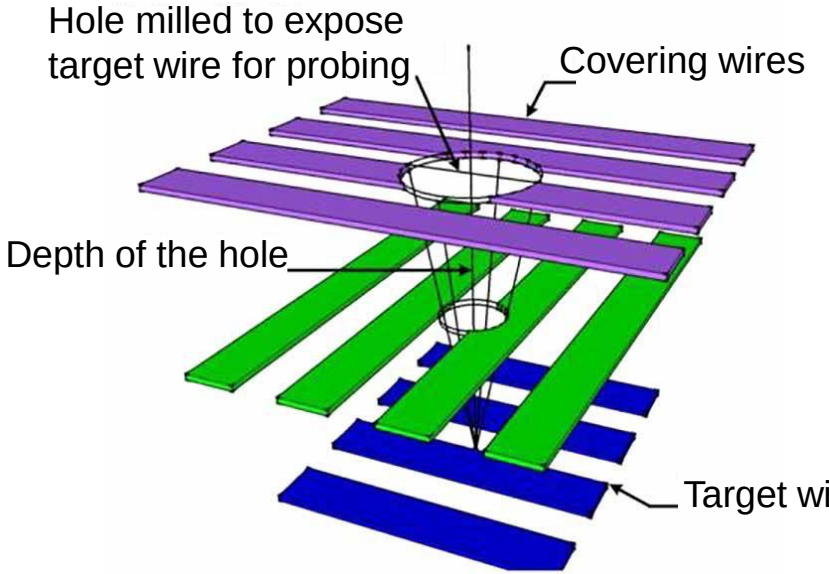


Illustration of the frontside probing attacks with FIB
[Wang et al., Design & Test'17]

Fault Injection Attack

- Deduce sensitive information by directly or indirectly injecting faults during the cryptographic operation
  - Direct fault injection
    - Ex: using laser light or electromagnetic waves
  - Indirect fault injection
    - Ex: repetitively writing to particular memory locations

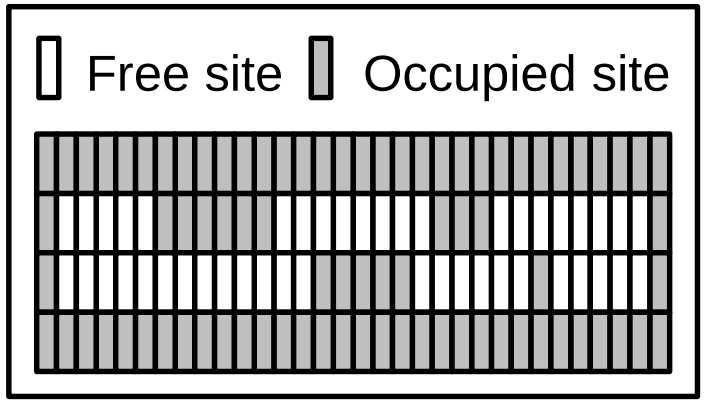
- This work concentrates on frontside direct fault injection
  - Ex: laser fault injection

Evaluation - Exploitable Region

- A region that can be used by attackers to insert Trojans
  - A set of spatially continuous sites that is larger than the Trojan-placement threshold (20)

- The routing resource of the exploitable regions is the number of free tracks above the exploitable regions
  - Evaluate the number of all tracks above the exploitable regions

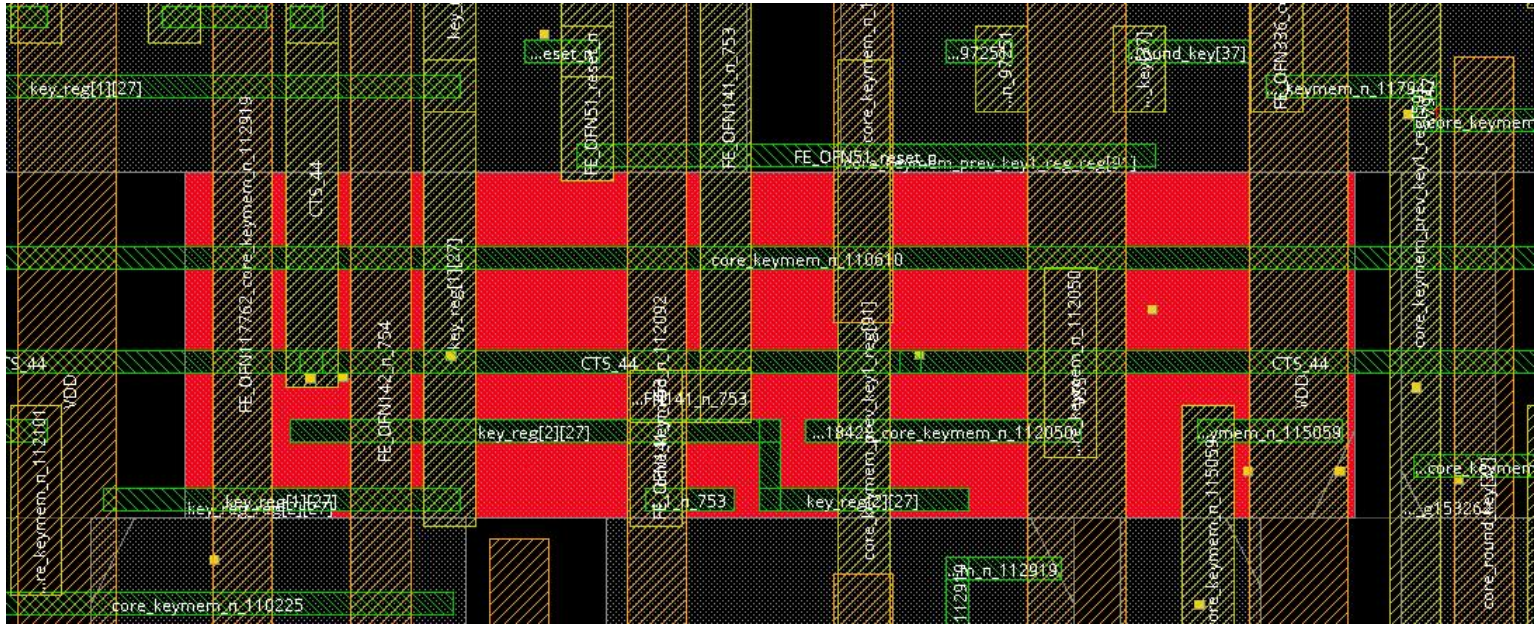
- Exploitable distance is the farthest distance among all legal Trojan-placement positions from the target assets
  - Estimate the slacks with information from the library
    - Ex: wiring loads and capacitance loads



A simple illustration of the exploitable region

Evaluation - Exposed Area

- The region of the cell and net assets is accessible by the probes from the frontside via the metal layers
  - Assume the probes are infinitely thin
  - Consider only the frontside attack at zero degrees
- Increasing exposed area provides an expanding attack surface
  - Attackers gain more chances to attack on the target assets



Exposed area

An example of the exposed area of a cell

Problem Formulation

- Given
  - Post-route DEF file (.def)
  - Post-route netlist file (.v)
  - Technology and cell library file (.lef)
  - Timing library (.lib, .sdc)
  - Cell assets (the list of sensitive cells to be protected)
  - Net assets (the list of sensitive nets to be protected)

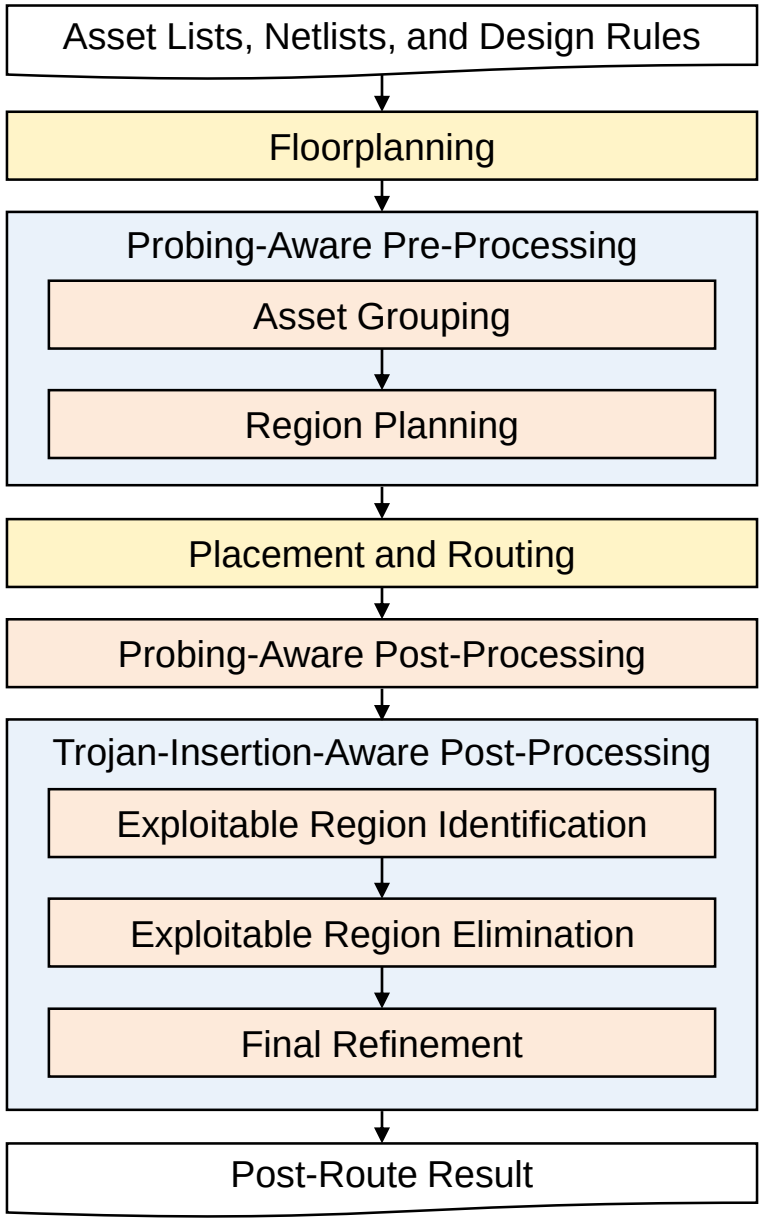
- Output
  - A valid post-route solution (.def, .v)

- Objective
  - Harden the physical layouts at design time against Trojan insertion, frontside probing, and fault injection attacks

Outline

- Introduction
- Preliminaries
- Proposed Framework
- Experimental Results
- Conclusions

Proposed Framework

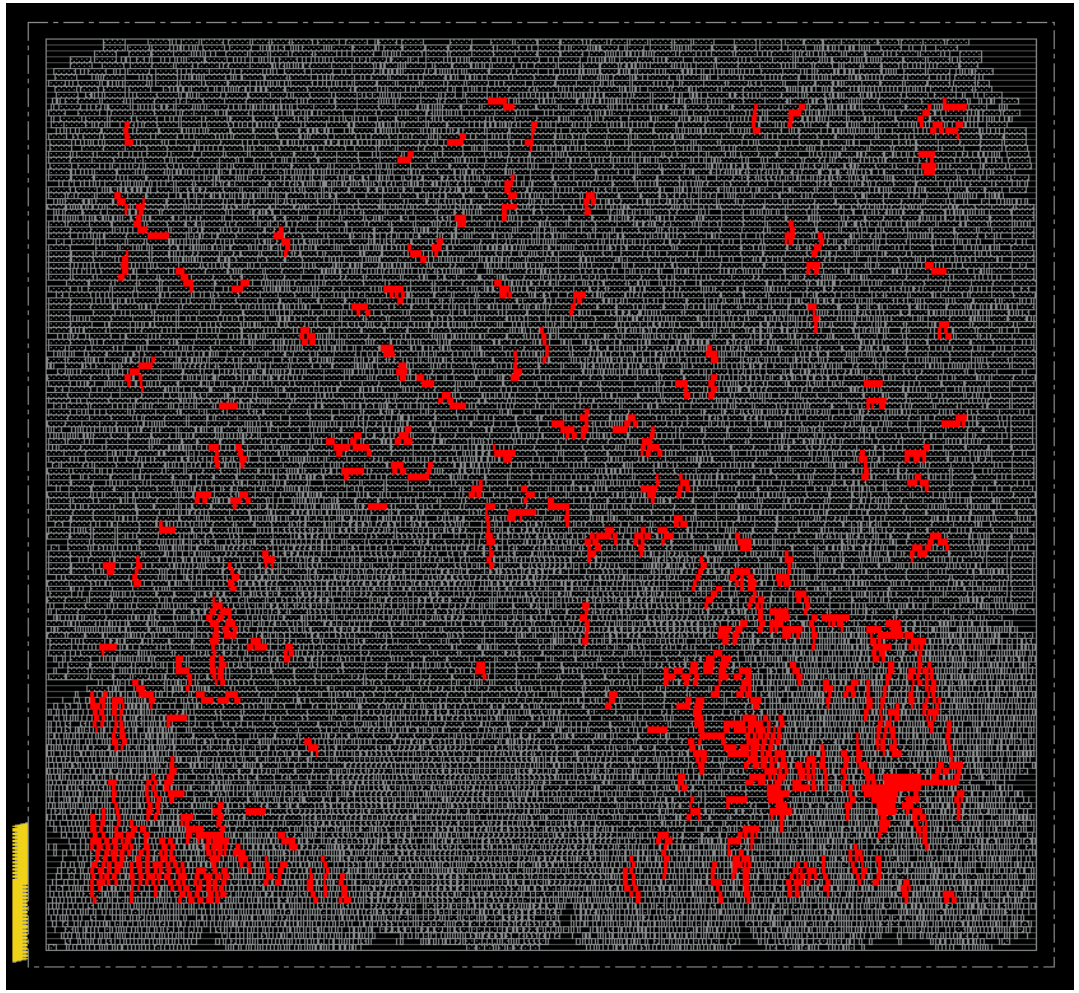


Floorplanning

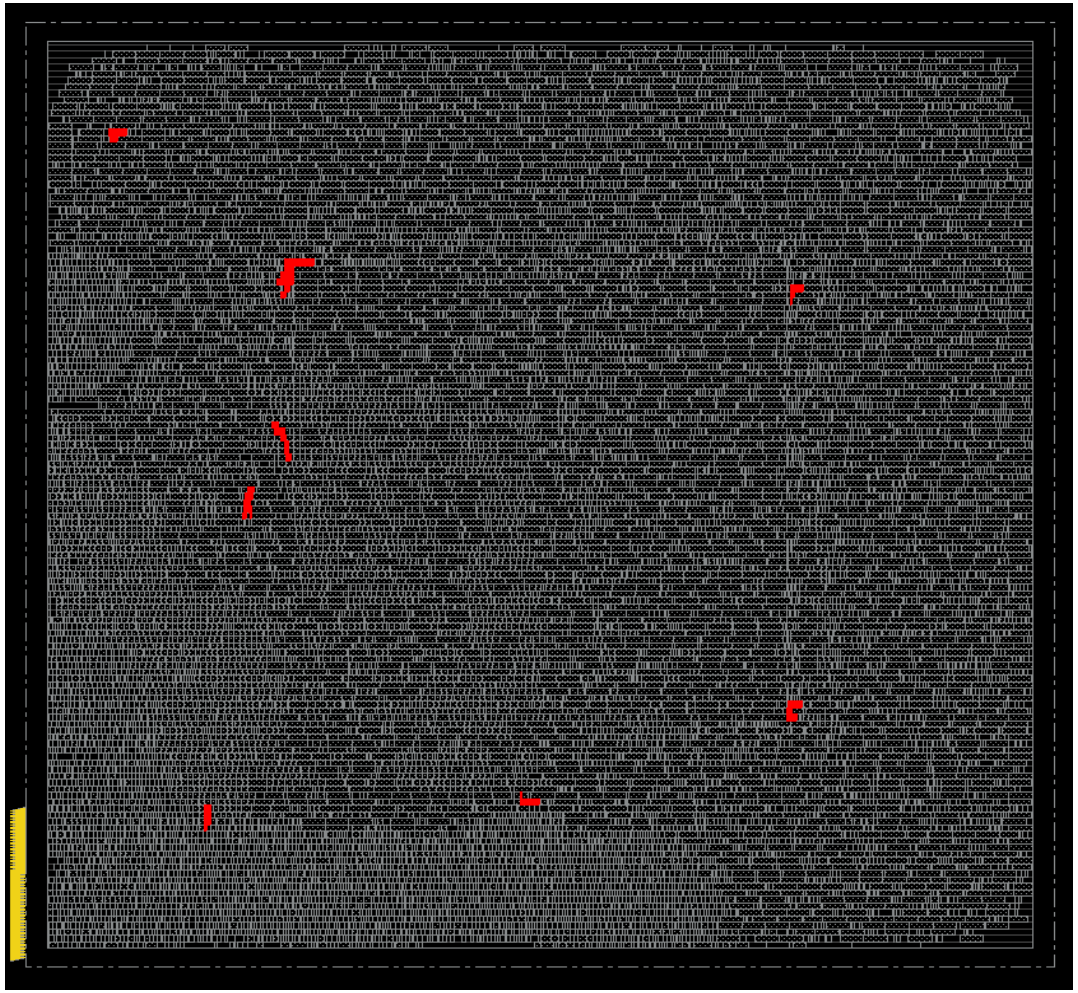
- Extract the PDN information from the original netlist
- Rescale the core area with high core utilization rate
  - Set the core utilization rate to 90%–95% based on the experiments to prevent timing closure failures
  - $core\ utilization = \frac{standard\ cells\ area}{core\ area}$
- Reconstruct the PDN
- Most modern EDA tools will perform netlist reconstruction at the placement stage to get better performance in timing and area metrics
  - Apply the "*dont_touch*" attribute to the cells and nets in the asset list
    - Cells that are buffers or inverters
    - Nets connected to these cells (buffers or inverters)

A Comparison of Layouts with Different Utilization Rates

Exploitable regions

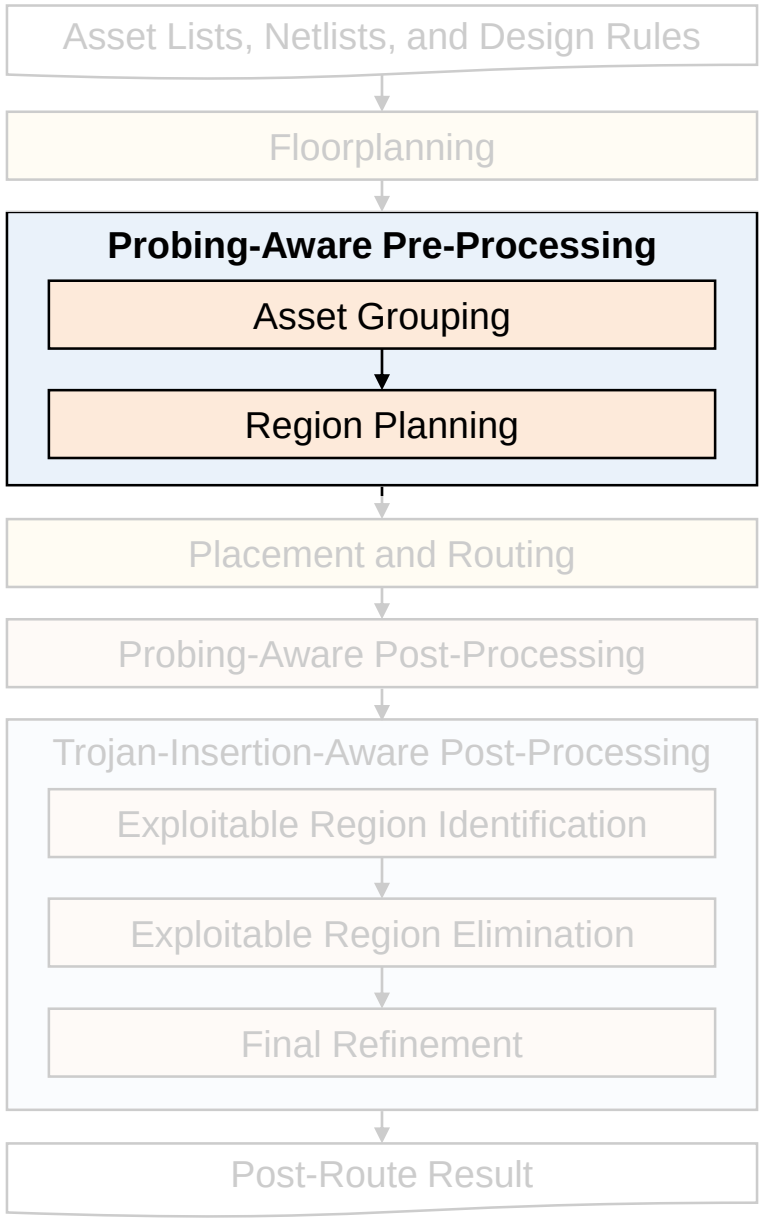


(a) A layout with a 75% utilization rate



(b) A layout with a 94% utilization rate

Proposed Framework



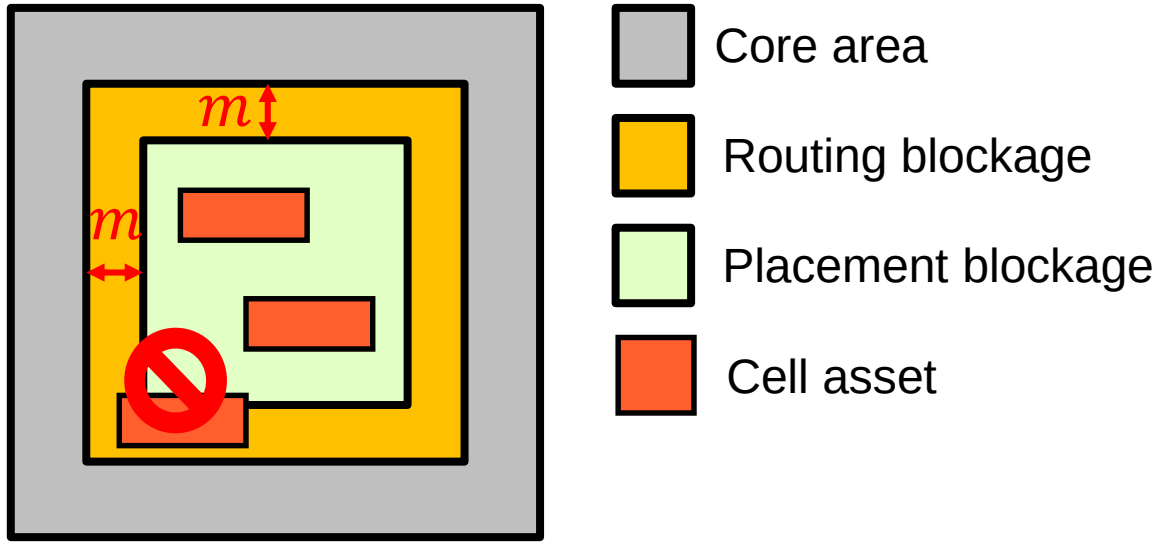
Asset Grouping

- Most state-of-the-art placers use routability-driven placement to address the congestion issues during placement
  - These techniques increase the separation between cells to preserve more routing resources
  - Protecting cell and net assets that are spread out all over the layout is complicated and inefficient

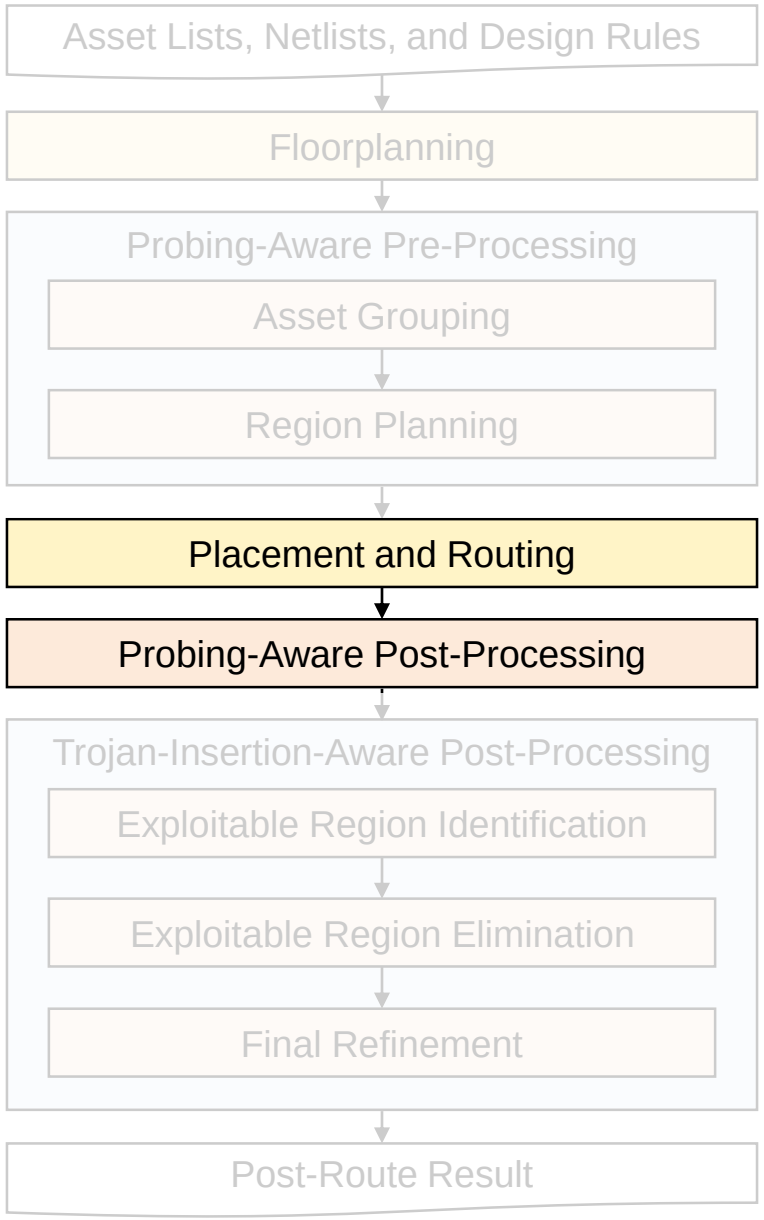
- Group both cell assets and cells connected to net assets
  - Assign a higher weight to net assets to make assets closer and shorter

Region Planning

- Construct a placement and routing blockage
- Placement blockage
  - A type of placement region constraint
  - Blockage size is based on the area ratio of the asset group area and the core area
- Routing blockage
  - Nets cannot be routed in this area
  - Set in the top metal layer
- m is set to the minimum value that satisfies a parallel run length spacing constraint

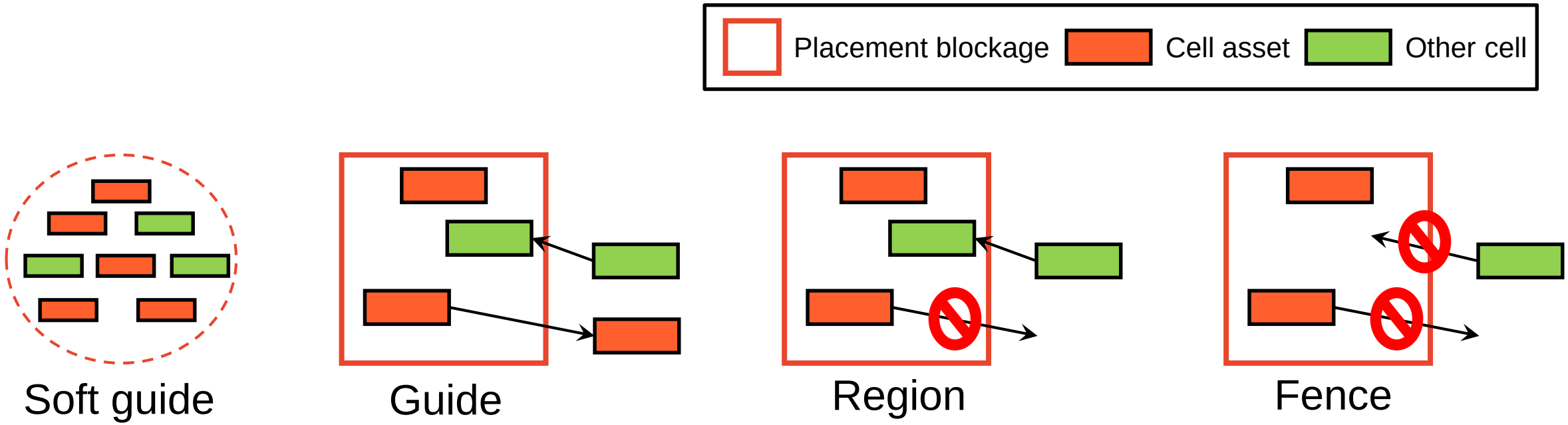


Proposed Framework



Placement and Routing

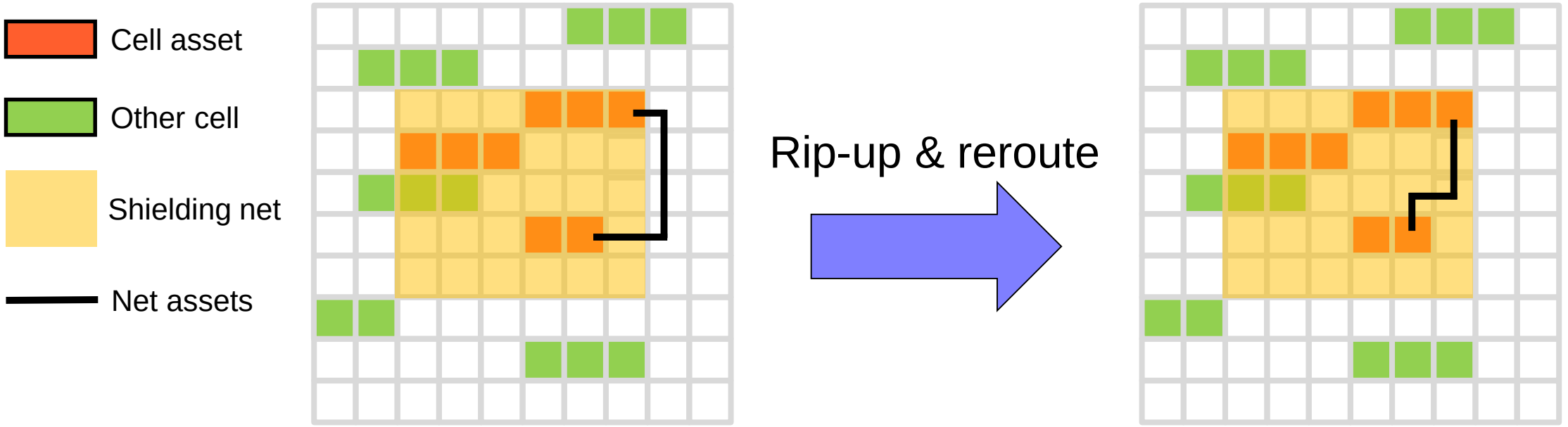
- Perform region-aware placement and routing
  - Any placer supports region constraints can also be used



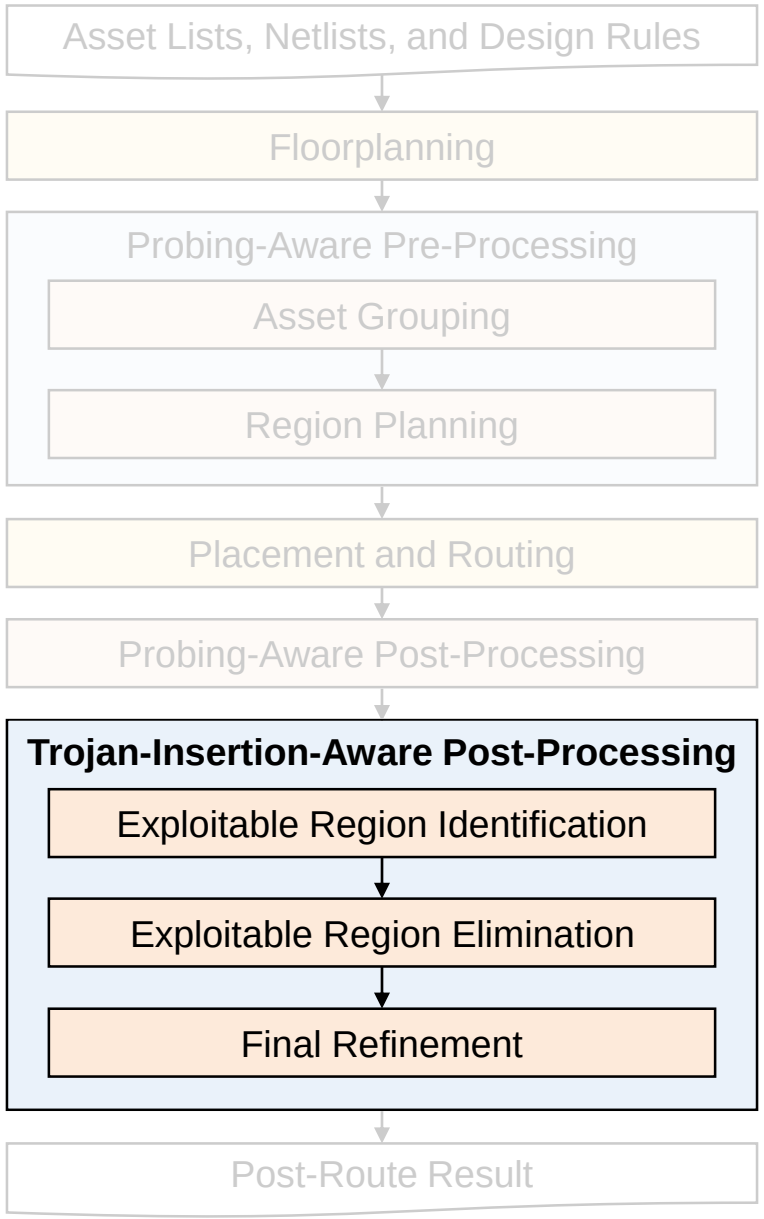
Probing-Aware Post-Processing

- Create a large-scale shielding net to cover all cell and net assets
  - Shielding net is in the top metal layer
  - The size of a shielding net is the same as the routing blockage

- Verify if all the net assets are routed within a routing blockage
  - Perform rip-up and reroute to ensure all net assets are routed inside the blockage

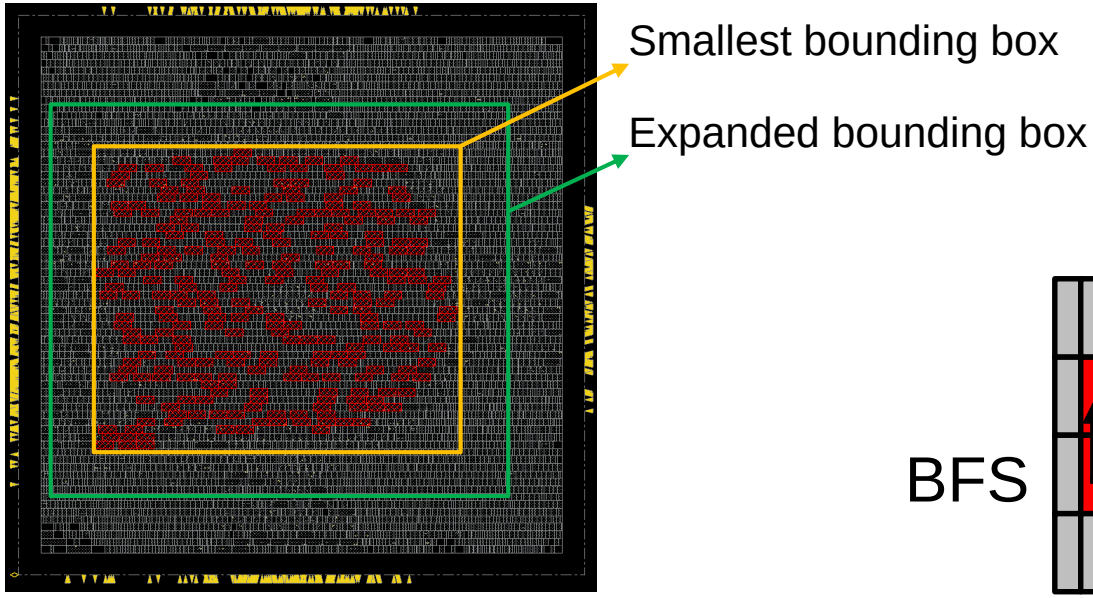


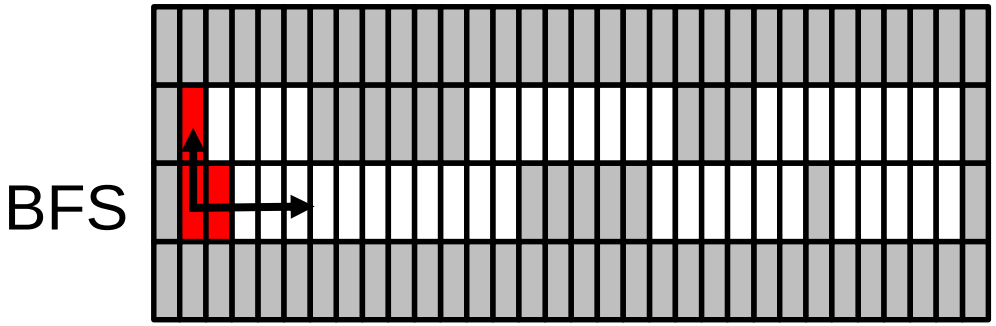
Proposed Framework



Exploitable Region Identification

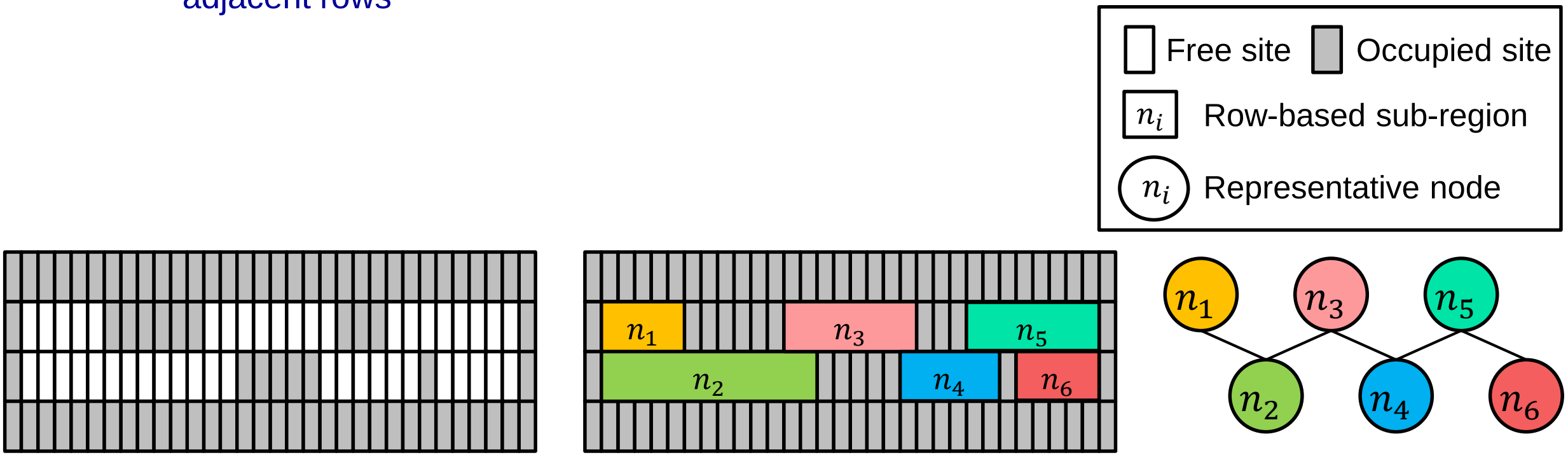
- Identify all exploitable regions to be removed
- Find the smallest bounding box covering all cell assets
  - Expand the bounding box by the value of exploitable distance
- Perform breadth-first search (BFS) for each row inside the bounding box





Row-based Graph

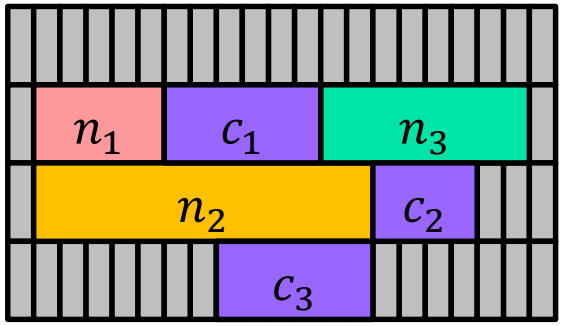
- Partition an exploitable region into sub-regions
  - Transform each sub-region into a node with weight equal to the number of sites in the sub-region
  - Edges are constructed to represent the connection of consecutive sub-regions in adjacent rows



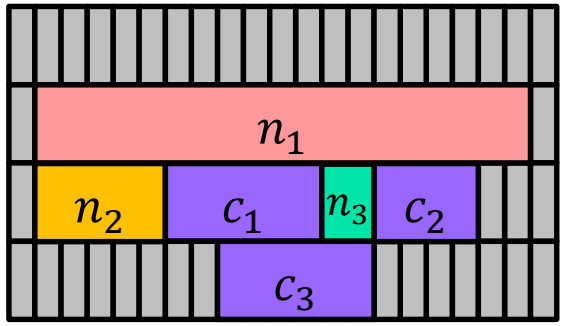
(a) An example of an exploitable region (b) Row-based sub-regions of the exploitable region (c) The graph transformed from (b)

Exploitable Region Elimination

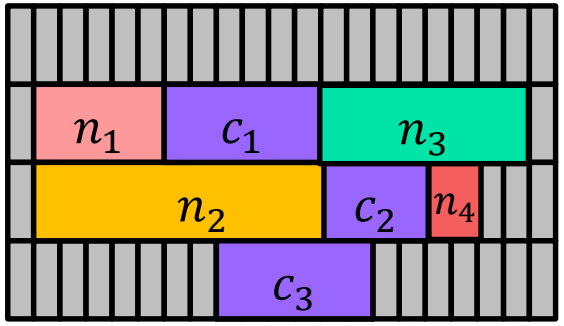
- Perform cell-movement-based method to eliminate the exploitable regions greedily
  - Exploitable regions are eliminated by separating them into regions with the number of continuous sites less than the Trojan-placement threshold



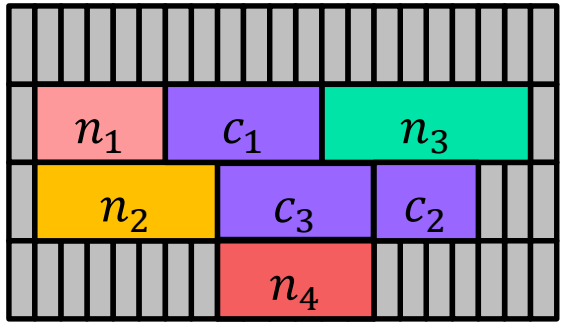
(a) The original placement



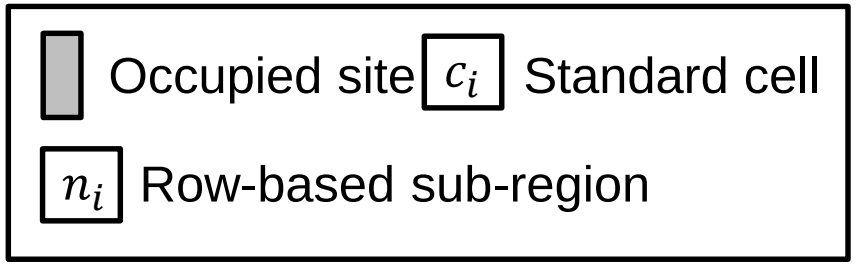
(b) c_1 is moved to cut n_2



(c) c_2 is moved to cut n_2



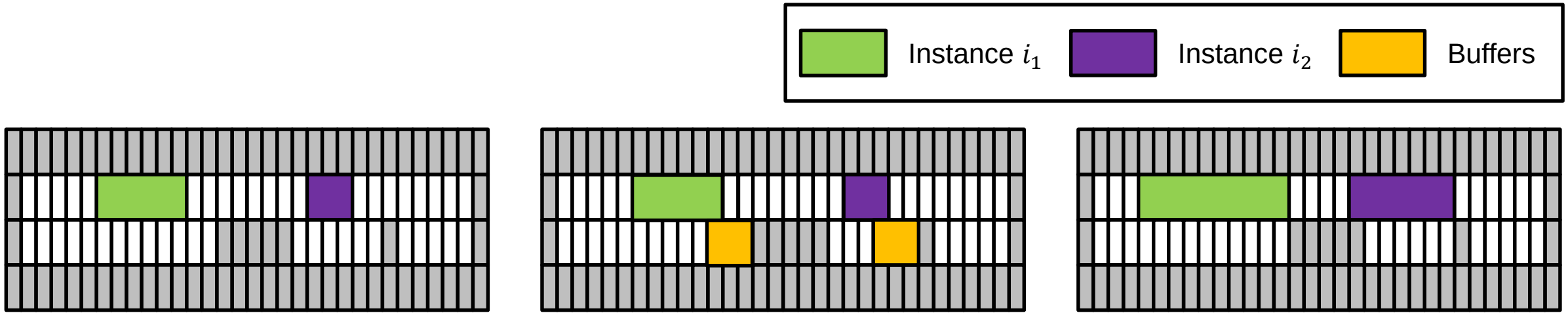
(d) c_3 is moved to cut n_2



Final Refinement

- If exploitable regions still exist
  - Standard cells around these exploitable regions are too small or too sparse to divide the exploitable regions through cell movement

- Upsizing nearby cells and/or inserting dummy buffers are applied to remove all remaining exploitable regions



(a) Original placement result

(b) Placement result after inserting buffers

(c) Placement result after upsizing cells

Outline

- Introduction
- Preliminaries
- Proposed Framework
- Experimental Results
- Conclusions

Benchmark Statistics

- All benchmarks are from the 2022 ACM ISPD Security Closure of Physical Layouts Contest
  - Synthesized by the *Synopsys Design Compiler* with the *Nangate 45nm Open Cell Library*

Benchmark	# cells	# cell assets	# nets	# net assets	Utilization	# Layers
AES_1	16509	336 (2.04%)	19541	468 (2.39%)	75.10%	10
AES_2	16509	336 (2.04%)	19541	468 (2.39%)	75.10%	10
AES_3	15836	322 (2.03%)	18868	461 (2.44%)	94.80%	10
Camellia	6710	265 (3.94%)	7094	384 (5.41%)	51.10%	6
CAST	12682	1886 (14.87%)	13050	1919 (14.70%)	51.30%	6
MISTY	9517	256 (2.68%)	9895	14 (0.14%)	51.60%	6
PRESENT	868	164 (18.89%)	1030	145 (14.07%)	50.60%	6
SEED	12682	4810 (37.92%)	13055	4938 (37.83%)	51.37%	6
TDEA	2269	168 (7.40%)	2592	168 (6.48%)	81.12%	6
SPARX	128	8146 (1.57%)	10786	2176 (20.17%)	50.99%	6
openMSP430_1	4690	1541 (32.85%)	5310	1393 (26.23%)	50.46%	6
openMSP430_2	5921	1839 (31.05%)	6307	1717 (27.22%)	80.09%	6

Comparison with The Top-3 Winner Teams

- All metrics are normalized to their respective nominal baselines
  - 0: maximum improvement, 1: minimum improvement
- Compared to the top-3 winner teams
  - Achieve a better average score with 0.5%, 8.1%, and 12.8% smaller design cost, respectively
  - Obtain the smallest design cost for 7 out of 12 benchmark circuits

Benchmark	1st place			2nd place			3rd place			Ours		
	des	ti	fsp_fi	des	ti	fsp_fi	des	ti	fsp_fi	des	ti	fsp_fi
AES_1	0.447645	0	0	0.475469	0	0	0.519014	0	0	**0.427583**	0	0
AES_2	**0.425056**	0	0	0.458233	0	0	0.509517	0	0	0.438185	0	0
AES_3	**0.473199**	0	0	0.498813	0	0	0.541594	0	0	0.485633	0	0
CAST	0.412035	0	0	0.409304	0	0	0.439133	0	0	**0.396717**	0	0
Camellia	**0.398203**	0	0	0.420739	0	0	0.418330	0	0	0.405498	0	0
MISTY	0.418306	0	0	0.396844	0	0	0.417127	0	0	**0.387017**	0	0
PRESENT	**0.359781**	0	0	0.427651	0	0	0.446817	0	0	0.363447	0	0
SEED	0.416061	0	0	0.442646	0	0	0.442510	0	0	**0.414805**	0	0
SPARX	0.397067	0	0	0.420406	0	0	0.404258	0	0	**0.387243**	0	0
TDEA	**0.459273**	0	0	0.526013	0	0	0.524128	0	0	0.484775	0	0
openMSP430_1	0.406426	0	0	0.440711	0	0	0.469361	0	0	**0.401875**	0	0
openMSP430_2	0.464010	0	0	0.543684	0	0	0.570014	0	0	**0.460260**	0	0
Average	0.423089	-	-	0.455043	-	-	0.475150	-	-	**0.421087**	-	-
Ratio	1.005	-	-	1.081	-	-	1.128	-	-	1.000	-	-

Outline

- Introduction
- Preliminaries
- Proposed Framework
- Experimental Results
- Conclusions

Conclusions

- We proposed a security-aware framework against Trojan insertion, frontside probing attacks, and fault injection attacks while minimizing extra design cost
  - A large-scale shielding method covers all the exposed cell and net assets
  - A cell-movement-based method eliminates the exploitable regions with low overheads

- Our proposed framework can achieve the best score among all the other teams in the 2022 ISPD contest, which shows the effectiveness of our work

Thank You!