

Physical-Aware eFPGA Redaction for Secure and Efficient Hardware IP Protection

Yunqi He, You Li*, Ruofan Huang, Guannan Zhao, and Hai Zhou

Northwestern University, Evanston, IL, USA

{yunqi.he, you.li, ruofanhuang, gnzhaoh}@u.northwestern.edu, haizhou@northwestern.edu

Abstract—Embedded FPGA (eFPGA)-based hardware redaction has emerged as a promising technique for protecting the intellectual property (IP) of integrated circuits. Existing approaches select a subset of the logic at the register-transfer level (RTL) and replace it with a programmable eFPGA module. However, due to their lack of awareness of physical information, these approaches incur significant power, performance, and area (PPA) overhead on the resulting chip. This paper presents a physically guided partitioning approach that divides the original design into two parts: one implemented as an application-specific integrated circuit (ASIC) and the other redacted onto an embedded FPGA fabric. It leverages a graph neural network to encode both the structural and physical information of each gate into an embedding vector. It then employs a clustering and selection process to identify the redaction candidate. Experiments demonstrate that our approach consistently reduces timing overhead while achieving comparable or superior results in terms of area, security, and resource consumption.

Index Terms—hardware security, IP piracy, embedded FPGA, logic locking, chip partitioning

I. INTRODUCTION

The semiconductor industry has shifted to a globalized business model in which the integrated circuit (IC) supply chain is outsourced to offshore manufacturing and testing facilities. Design houses are facing mounting threats in terms of IP piracy, reverse engineering, and overproduction [1]. According to a commission report by NBR [2], hardware IP theft has caused an estimated financial loss of hundreds of billions of US dollars to the industry.

Researchers have developed various techniques to enhance the supply-chain security of ICs. Logic locking [3]–[5] inserts key-controlled locking units to protect intellectual property and resist security threats. Without knowing the correct key, an adversary cannot recover the functionality of the original circuit. IC camouflaging [6] disrupts the adversary’s capability to extract the netlist of a circuit from imaging its physical layout. Split manufacturing [7] divides the fabrication process across multiple foundries, with lower metal layers manufactured by the untrusted foundries. However, all of these techniques are susceptible to I/O query attacks [8], [9], structural attacks [10], [11], or proximity attacks [12]–[14]. Moreover, they rely on special security primitives and incur significant PPA overheads, preventing them from being widely adopted by the industry.

* Corresponding author.

This work is partially supported by the National Science Foundation under grants 2113704 and 2148177.

Recently, eFPGA-based hardware redaction [15] has emerged as a more robust and efficient approach to safeguard the supply chain of ICs. During the design process, the design house replaces a small portion of the original circuit with an embedded FPGA fabric. Upon completion of the manufacturing process, the design house configures the eFPGA with the correct bitstream to activate the chip’s functionality. There exist effective ways to prevent the bitstream from being tampered with by an adversary [16].

A central problem of eFPGA redaction is how to partition the original circuit into two portions: the modules remain as ASIC, and the remaining modules are to be redacted into an FPGA. A desirable partitioning result can minimize the PPA overhead of the resulting chip (in this work, we focus on delay and area, as power typically correlates with area) while maintaining the resiliency against I/O query attacks. Previous work, such as *SheLL* [17] and the *ALICE* framework [18], along with its successor *ARIANNA* [19], tackle this problem primarily at the RTL. However, they cannot find the best trade-off between performance, security, and cost of redaction because they are unaware of the structural and physical information of the chip.

To address these limitations, this paper proposes a physically-aware eFPGA redaction approach that leverages both structural and physical information to make optimal partitioning decisions. Unlike existing RTL-level methods that operate without knowledge of the final physical implementation, our approach performs complete synthesis and place-and-route to extract precise timing, spatial, and connectivity information. This physical awareness enables more intelligent partitioning decisions that minimize PPA overhead while maintaining strong security guarantees.

The main contributions of this paper are fourfold:

- We identify the core drawback of existing redaction techniques: overlooking placement and timing information during redacted region selection, which results in high PPA overheads.
- We propose a method to incorporate both structural and physical information into a heterogeneous graph.
- We leverage self-supervised GNN training followed by k -means clustering to keep adjacent cells with similar timing, structural, and physical metrics in the same group.
- We develop a multi-objective group selection strategy to achieve a desirable trade-off among PPA overheads, attack resilience, and resource utilization.

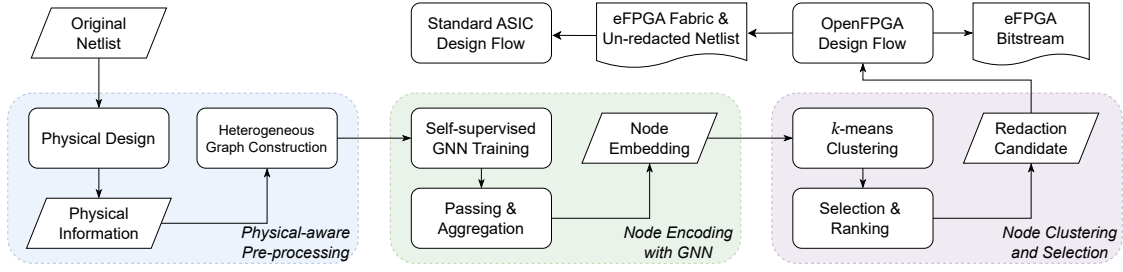


Fig. 1: General workflow of the proposed eFPGA redaction method.

- We conducted extensive experiments across various benchmarks and technologies to demonstrate the consistent superiority of the proposed method.

The remainder of this paper is organized as follows. Section II presents the background and problem definition. Section III details our physical-aware redaction methodology, including graph construction, GNN training, node clustering, and group selection algorithms. Section IV presents experimental results and analysis. Finally, Section V concludes the paper and outlines future research directions.

II. BACKGROUND AND PROBLEM FORMULATION

A. eFPGA-based Hardware Redaction

An FPGA is a reconfigurable circuit capable of implementing different functionalities through a *bitstream*. An embedded FPGA (eFPGA) is a complete IP core that can be integrated into a chip alongside other components. Only entities possessing the eFPGA bitstream can recover the chip’s intended functionality, making it an effective tool for IP protection. A design house can exploit this feature by replacing a portion of the original circuit with an eFPGA, preventing adversaries in the IC supply chain from recovering the chip’s functionality. Once the chip returns to the design house, the eFPGA is programmed with the bitstream to restore its original behavior. Compared with standard logic locking, eFPGA redaction is more resilient to I/O attacks, as the interconnections and lookup tables (LUTs) are inherently more complex than a simple vector of key bits. Moreover, eFPGA redaction does not require a dedicated key storage module, since the secret is encoded directly in the bitstream.

ALICE [18] and SheLL [17] are two representative methods for eFPGA redaction. ALICE introduces an automated flow to select crucial modules for redaction at the RTL level. It identifies modules based on their impacts on designated outputs, merges small independent modules, and makes selections based on hardware cost and security level. This approach is further refined in the ARIANNA framework [19], which extends ALICE by optimizing the eFPGA fabric parameters to reduce hardware overhead. In contrast, SheLL selects the interconnects between modules and their associated cells for redaction to minimize redaction overhead.

B. GNN-based VLSI Physical Design Automation

The aforementioned redaction techniques focus on RTL or structural levels and overlook physical design information. In

contrast, recent advances in GNNs enable the joint modeling of structural and physical features, leading to more effective optimization. An early effort is TP-GNN [20], which leverages unsupervised graph learning to solve tier partitioning for monolithic 3D ICs. Subsequent studies have explored the use of GNNs across multiple stages of the design flow: Lu et al. [21] employed a graph framework for placement optimization, achieving improvements in wirelength and timing; Xie et al. [22] introduced Net2, a graph attention model tailored for pre-placement net-length estimation; Guo et al. [23] developed a timing-engine-inspired GNN for pre-routing slack prediction; a recent work has explored GNN-based approaches for timing analysis [24]. At a higher level of abstraction, Liu et al. [25] proposed GraphPlanner for floorplanning, while Ferretti et al. [26] applied GNNs to high-level synthesis exploration. Collectively, these studies demonstrate that circuit graphs augmented with physical features can effectively guide performance-driven design decisions. Inspired by these advances, we extend graph learning to the security domain and introduce a GNN-based framework for eFPGA redaction.

C. Attack Model and Security Considerations

The primary threat to eFPGA redaction is the I/O attack. If an adversary gains access to an activated chip, it can query the eFPGA part through the scan chain and observe the corresponding outputs. In a SAT-based attack, the bitstream is treated as a key vector, and the input–output pairs are used to solve for the key [27]. Alternatively, approximate logic synthesis techniques can be employed to reconstruct the bitstream [28]. In practice, both approaches become inapplicable when the bitstream exceeds 4,000 bits.

D. Problem Definition

Given a circuit netlist $N = (V_g, V_n)$ where V_g represents the gate set and V_n represents the net set, the goal of an eFPGA redaction algorithm is to construct a partition $\Pi : N \rightarrow \{N_{ASIC}, N_{eFPGA}\}$, such that the PPA overheads after redaction are minimized, resilience against eFPGA I/O attacks is ensured, and all eFPGA resource constraints are satisfied. Existing RTL-level approaches cannot fully capture physical design information, often resulting in suboptimal partitioning.

III. PHYSICAL-AWARE EFPGA REDACTION FLOW

A. Overview

The general design flow of the proposed eFPGA redaction method is shown in Fig. 1. It consists of the following stages:

(1) *Extracting physical information.* Starting from an RTL design, we perform logic synthesis and physical design to generate a physical layout. As such, we can extract the detailed physical information for each gate.

(2) *Constructing heterogeneous graph.* We construct a *heterogeneous graph* that represents the circuit with various nodes and edges. Each node contains coordinates, connectivity, and timing information of the corresponding *gate* or *net*.

(3) *Training GNN encoder.* We use a self-supervised learning strategy to train a GNN encoder. Specifically, we perform *contrastive learning* to generate an embedding vector for each node to capture the essential structural and physical information. During training, each node aggregates information from its neighbors to update its own representations.

(4) *Gate clustering.* We perform k -means clustering to partition the design into disjoint groups. Each group comprises gates that have similar embeddings and are connected to each other.

(5) *Selecting group.* We evaluate the groups according to their structural, physical, I/O, and security characteristics. The one satisfying all design constraints and achieving the best result is selected to be redacted to the eFPGA.

B. Graph Construction and Feature Engineering

We construct a heterogeneous graph $G = (V, E)$ to model the netlist of a circuit. We create a *gate node* for each standard cell and a *net node* for each net. We create two edges $(g, n), (n, g) \in E$ between a gate node $g \in V_g$ and a net node $n \in V_n$, if they are connected in the original netlist. These operations yield a bipartite graph, whose nodes are divided into two disjoint sets V_g and V_n . This graph preserves the topology of the original netlist.

In addition to structural information, we perform physical design to obtain physical information. Then we annotate each node with the following features:

- *Structural features.* These include cell type (as a one-hot vector), connectivity to primary inputs and outputs, fan-in degree, and fan-out degree.
- *Timing feature.* We use the slack t_g to represent the criticality of a gate. In practice, we use OpenSTA [29] to collect maximum delay paths between pairs of registers and update t_g of each gate to the worst value.
- *Physical features.* These include x and y coordinates in the physical layout, distance to the layout boundaries, and the half-perimeter wirelength of the net driven by the node.

All features are normalized to foster effective learning. In particular, x and y coordinates are normalized by the width and height of the layout.

C. Network Configuration

To facilitate the downstream task, we employ a GNN to generate an embedding vector \mathbf{h}_g for every gate node. Specifically, we construct a heterogeneous graph attention network (GAT) [30] (Fig. 2, left) that takes a heterogeneous graph as input and adopts the following architecture:

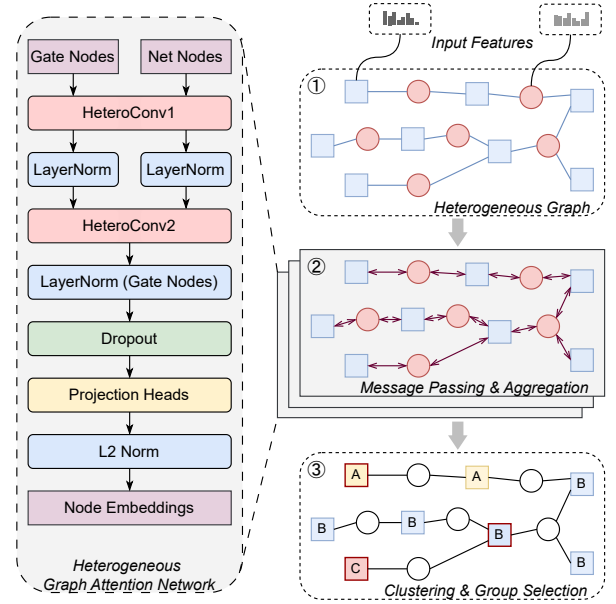


Fig. 2: GNN-based redacted region selection.

- *Layer 1:* A multi-head GATConv (Graph Attention Convolution) component is applied to every heterogeneous edge, *i.e.*, $g \rightarrow n$ or $n \rightarrow g$. As such, the network can focus on the most relevant features for each node and learn different weights for different connections. A HeteroConv component is applied to manage message passing and aggregation between nodes. The output dimension is $H \times h$, where H is the number of heads and h is the size of the intermediate node representation.
- *Layer 2:* It is similar to *Layer 1*, except that we choose single-head GATConv to reduce the output dimensionality. Therefore, the output dimension for each node is h .
- *Layer 3:* It is a 2-layer multi-layer perceptron with *ReLU* as the activation function, followed by a ℓ_2 -normalization layer. It outputs a d -dimension embedding vector for every gate node.

D. Loss Function

Due to the absence of ground-truth labels, we solve the redaction problem with unsupervised learning. The loss function consists of three terms:

$$\mathcal{L}_u = \mathcal{L}_s + \mathcal{L}_b + \mathcal{L}_t, \quad (1)$$

where \mathcal{L}_u denotes the total loss of the unsupervised learning task, \mathcal{L}_s denotes the similarity loss, \mathcal{L}_b denotes the boundary loss, and \mathcal{L}_t denotes the timing loss.

- *Similarity loss* [31] [32] [33]: This term is expected to reach its minimum when adjacent gates have similar embeddings and distant gates have different embeddings. Consequently, gates within the same area have a high probability of being assigned to the same group. Algorithm 1 summarizes our approach to prepare training data for each gate node. It enumerates all edges in the heterogeneous graph and collects both positive

Algorithm 1 Collect Training Data for Similarity Loss.

Require: A heterogeneous graph $G = (V_g \cup V_n, E)$

Ensure: A set of training data \mathcal{D} . Each sample is a triple (g, pos, \mathcal{N}) , where $g \in V_g$ is an anchor, $pos \in V_n$ is a positive neighbor, and $\mathcal{N} \subseteq V_n$ is a set of negative neighbors.

```
1:  $\mathcal{D} \leftarrow \emptyset$ 
2: for all  $edge(g, n) \in E$  do       $\triangleright$  for every  $(gate, net)$  pair
3:    $pos \leftarrow n$                    $\triangleright$  positive net
4:   Sample  $w_s \in V_n$  s.t.  $(g, w_s) \notin E$   $\triangleright$  soft negative net
5:   Sample  $w_h \in V_n$  s.t.  $(g, w_h) \notin E$ 
       $\wedge \deg(w_h) \approx \deg(n)$            $\triangleright$  hard negative net
6:    $\mathcal{N} \leftarrow \{w_s, w_h\}$ 
7:    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(g, pos, \mathcal{N})\}$ 
8: end for
9: return  $\mathcal{D}$ 
```

and negative examples as training data. A positive example (g, pos) corresponds to an edge in the original graph. A soft negative example (g, w_s) is a random pair where the two nodes are not adjacent in the original graph. A hard negative example (g, w_h) further requires that w_h matches pos in some designated attributes, such as net size and slack. The similarity loss is defined as

$$\mathcal{L}_s(g, pos) = -\log \frac{\exp(\mathbf{h}_g \cdot \mathbf{h}_{pos} / \tau)}{\exp(\mathbf{h}_g \cdot \mathbf{h}_{pos} / \tau) + \sum_{w \in \mathcal{N}} \exp(\mathbf{h}_g \cdot \mathbf{h}_w / \tau)}. \quad (2)$$

The temperature constant τ scales the logits before the softmax, thereby controlling the sharpness of positive pairs over the negative ones.

- **Boundary loss:** I/O resources are usually the bottlenecks of eFPGA redaction. We design a simple term to penalize excessively large cut sizes of groups after clustering. Specifically, we collect a set of random positive and negative training examples, and use the cross-entropy loss:

$$\mathcal{L}_b = - \sum_{(g, w) \sim E} \log(\sigma(\mathbf{h}_g \cdot \mathbf{h}_w)) - \sum_{(g, w) \not\sim E} \log(1 - \sigma(\mathbf{h}_g \cdot \mathbf{h}_w)), \quad (3)$$

where σ denotes the *sigmoid* function. Notice that we do not aim for a minimum input size, as doing so may reduce robustness against I/O attacks. Compared to the spectral clustering method [34], the boundary loss is substantially easier to compute.

- **Timing loss:** The speed of an eFPGA is typically much slower than that of an ASIC. To account for the timing difference between the two types of hardware, we need to ensure that all gates in the redacted group have sufficiently large positive slacks. This approach allows the chip after redaction to achieve optimized performance. Formally,

$$\mathcal{H} = \{v \in V_g \mid s_v \geq \theta\} \quad (4)$$

represents the set of gates that have high positive slacks above a threshold θ and are therefore insensitive to timing variations.

We expect the embedding vectors of these gates to be close to each other. Hence, we draw a set of random pairs \mathcal{P} from \mathcal{H} and penalize the opposite with the following term:

$$\mathcal{L}_t = \frac{1}{|\mathcal{P}|} \sum_{(a, b) \sim \mathcal{P}} \|\mathbf{h}_a - \mathbf{h}_b\|_2^2. \quad (5)$$

E. Clustering and Group Selection

Given the embedding vectors generated by the GNN, we apply unsupervised clustering to partition the original design into disjoint groups, ensuring that all gates within the same group have similar embeddings. We use the standard k -means algorithm [35], setting k slightly larger than the ratio of the total number of gates to the eFPGA capacity. As a result, most of the groups after clustering can be accommodated in the eFPGA. Afterward, we select a suitable group according to the following criteria:

- **Group size.** The group size $|G_i|$ represents the number of gates within a group G_i .
- **Bitstream size.** B_i denotes the length of eFPGA bitstream. We require $B_{\min} \leq B_i \leq B_{\max}$, so that the group is suitable for the eFPGA and is sufficiently large to resist I/O attacks.
- **I/O size.** Input size I_i denotes the number of primary inputs within the fan-in cone of G_i . Output size O_i denotes the number of primary outputs within the fan-out cone of G_i .
- **Security score.** The security score measures the I/O attack complexity on a group:

$$Q_i = I_i + \log(O_i + 1). \quad (6)$$

We let I_i dominate Q_i because the attack complexity is mainly determined by the input size [36]. We set a minimum threshold Q_{\min} and require $Q_i > Q_{\min}$.

- **Slack score.** The slack score S_i is the worst slack among all gates within a group, i.e., $S_i = \min_{g \in G_i} t_g$. We require $S_i \geq S_{\min}$ so that all gates within a group have sufficiently large positive slacks.
- **Boundary ratio.** For a group G_i , the boundary ratio ρ_i is defined as the number of out-connecting edges divided by $|G_i|$. It estimates the intensity of I/O resource usage. We set a maximum threshold ρ_{\max} and require $\rho_i \leq \rho_{\max}$.

We exclude all groups that violate any of the above constraints. We rank the remaining groups according to a score function, which is elaborated in Section IV-C.

IV. EVALUATIONS

A. Experimental Setup

We implement the proposed eFPGA redaction framework in PyTorch, using PyTorch Geometric [37] for GNN construction and training. All experiments are performed on a Linux laptop equipped with 32 GB of RAM, an Intel Ultra 185H processor, and an NVIDIA RTX 4090 Mobile GPU. We set a time limit of 12 hours for each SAT-based attack.

We evaluate our method on a set of open-source designs from the OpenROAD repository [29]. They include cryptographic modules (aes), RISC-V processor cores (ibex, riscv32i), multimedia codec (jpeg), and network controller

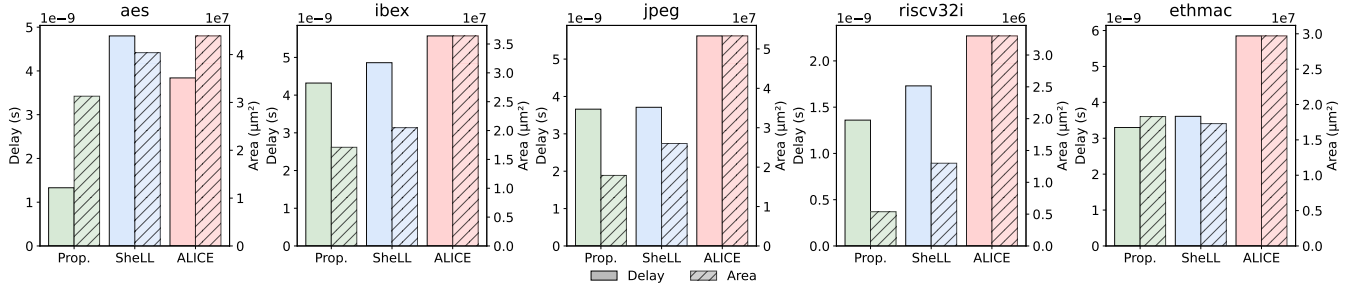


Fig. 3: Comparing area and delay overheads introduced by different redacted region selection methods.

(ethmac). The main characteristics of these benchmark circuits are summarized in Table I. To foster a fair comparison with existing work, we report the number of modules, the number of module instances, the range of I/O pin counts across all modules, and the total cell count.

TABLE I: Characteristics of benchmark circuits.

Design	#Modules	#Instances	#I/O pins	#Cells
aes	4	22	(16, 388)	15057
ibex	24	32	(4, 873)	15874
jpeg	83	148	(25, 784)	60745
riscv32i	25	66	(4, 223)	4698
ethmac	37	65	(6, 402)	58747

B. Baseline Selection

We compare the proposed redacted region selection method with two representative eFPGA redaction techniques. *Module-level redaction* such as ALICE leverages design engineer’s expertise [15] or an automated workflow [18] to identify RTL modules for redaction. In our experiments, we prioritize modules with critical functionalities and clean boundaries. *Routing-centric redaction* such as SheLL [17] instead selects the interconnections between modules as redaction candidates. In our experiments, we prioritize critical interconnection nets and their neighboring cells in adjacent modules. Each row in Table II presents the selected redacted region along with its coverage ratio, measured as a percentage of the total cell count of benchmark circuit.

C. Implementation Details

(1) *Feature extraction.* We use OpenROAD to perform preliminary physical design, enabling the extraction of desired physical information. Across all benchmarks, this process takes between 0.22 and 2.56 hours, with an average runtime of 1.64 hours. This is a one-time investment for each benchmark. To demonstrate the generality of the proposed method, we employ different standard cell libraries: Nangate 45nm [38] for aes, ibex, and jpeg; SkyWater 130nm [39] for riscv32i; and ASAP7 [40] for ethmac.

(2) *GNN training.* We set the number of attention heads to $H = 4$, the intermediate dimension to $h = 128$, the temperature constant to $\tau = 0.5$, and the dropout rate to 0.2. The batch sizes for the similarity, boundary, and timing losses

TABLE II: Redacted regions and their coverage ratios.

Design	Method	Redacted Region	Cov(%)
aes	Prop.	group 2	3.96
	SheLL	top-level AES datapath routing nets	7.32
	ALICE	key-gen block <i>aes_key_expand_128</i>	6.80
ibex	Prop.	group 76	1.15
	SheLL	if-id routing nets	1.14
	ALICE	control and status register block <i>ibex_csr</i>	1.00
jpeg	Prop.	group 60	1.16
	SheLL	<i>dct</i> → <i>quant</i> + <i>zigzag</i> → <i>entropy</i> routing nets	1.76
	ALICE	quantization divide unit <i>div_uu</i>	1.65
riscv32i	Prop.	group 1	10.94
	SheLL	if-id routing nets	11.24
	ALICE	decode/control block <i>controller</i>	17.03
ethmac	Prop.	group 72	1.17
	SheLL	RX/TX interface routing nets	0.90
	ALICE	<i>eth_wishbone</i> / <i>register</i> block	1.42

are 512, 1024, and 256, respectively. When computing the total loss, we assign a weight of 0.5 to the similarity loss and 0.25 to both the boundary and timing losses. Optimization is performed using Adam with a learning rate of 0.001. The GNN model is trained for 200 epochs, with early stopping applied if no improvement is observed over 20 consecutive epochs.

(3) *Clustering and group selection.* Once a group G_i satisfies the constraints in III-E, it is retained and assigned a composite *selection score*:

$$\text{Score}_i = \alpha B_i^{\text{norm}} + \beta S_i^{\text{norm}} + \gamma Q_i^{\text{norm}}, \quad (7)$$

where

$$B_i^{\text{norm}} = \min\left(1, \frac{\text{est_bits}_i}{\text{bits_thresh}}\right). \quad (8)$$

B_i^{norm} linearly scales the estimated configuration bits (est_bits_i) by the bit threshold (bits_thresh); S_i^{norm} represents the slack score of the group, normalized by the maximum across all groups; similarly, Q_i^{norm} denotes the normalized security score. All retained groups are then ranked in descending order of Score_i , and the top candidate is selected for eFPGA redaction. We set the score weights to $\alpha = 2.0$, $\beta = 3.0$, and $\gamma = 1.0$. Since all groups already satisfy the security threshold $Q_i^{\text{norm}} \geq Q_{\min}$, the relatively small weight on γ ensures that the final ranking emphasizes minimizing area and timing overhead (via B_i^{norm} and S_i^{norm}) rather than further prioritizing security.

TABLE III: Ablation study on AES and IBEX. Arrows indicate the direction of change (\uparrow increase, \downarrow decrease, $-$ unchanged).

Component Removed	AES								IBEX							
	PPA / SAT Security				Selection Scores				PPA / SAT Security				Selection Scores			
	Area	Delay	Security	Boundary	B_i	S_i	Q_i		Area	Delay	Security	Boundary	B_i	S_i	Q_i	
None	-	-	✓	-	-	-	-		-	-	✓	-	-	-	-	
Timing Feature	-27.3%	+150.1%	✓	$\uparrow\uparrow$	$\uparrow\uparrow$	\uparrow	\uparrow		+27.0%	+32.5%	✓	\uparrow	\downarrow	$\downarrow\downarrow$	-	
Physical Feature	-2.8%	+1.0%	×	\downarrow	\downarrow	\uparrow	\downarrow		+5.1%	-3.0%	✓	\uparrow	\uparrow	\downarrow	-	
Timing Loss	-3.6%	+253.9%	✓	\uparrow	$\uparrow\uparrow$	\uparrow	\downarrow		+30.0%	+1.0%	✓	\uparrow	\uparrow	\downarrow	-	
Boundary Loss	-19.2%	+188.1%	✓	\uparrow	\uparrow	\uparrow	\downarrow		+25.1%	+21.9%	✓	\uparrow	\uparrow	\downarrow	-	
Security Score	N/A	N/A	×	$\downarrow\downarrow$	$\downarrow\downarrow$	\uparrow	\downarrow		N/A	N/A	×	$\downarrow\downarrow$	$\downarrow\downarrow$	\uparrow	\downarrow	

(4) *eFPGA synthesis and generation.* We use OpenFPGA [41] together with its default 40nm library for FPGA synthesis and eFPGA fabric generation. The eFPGA fabric inherently integrates a scan chain, which allows scan-based testing even without programming a bitstream.

D. Results

Figure 3 compares the delay and area overheads incurred by different redacted region selection techniques across all benchmarks in Table. I. The proposed method consistently achieves the shortest critical paths. For example, for *aes*, the critical path is reduced to 1.33ns, compared to 4.80ns for *SheLL* and 3.84ns for *ALICE*. For *riscv32i*, the proposed approach shortens the critical path by up to $1.8\times$ relative to the baselines. For *ibex*, it simultaneously achieves the shortest delay and the smallest area. Even for designs where *SheLL* is competitive (*jpeg* and *ethmac*), our method still provides a slight timing advantage. In terms of total area, the proposed method achieves the lowest overhead for *aes*, *ibex*, *jpeg*, and *riscv32i*, while remaining close to the best alternative for *ethmac*. In contrast, *SheLL* can occasionally achieve a smaller area (*ethmac*) but at the cost of longer delays. Overall, these results demonstrate that the proposed method effectively balances timing and area, outperforming existing techniques across all evaluated benchmarks.

For all benchmarks and redaction techniques, the SAT-based attack (Section II-C) always reaches the time limit. While all techniques offer guaranteed security, the proposed method stands out by achieving the most favorable trade-off across the various design objectives.

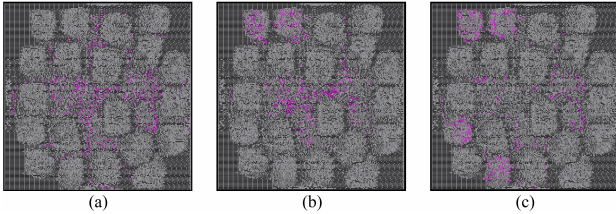


Fig. 4: Redacted regions selected by (a) the proposed method, (b) *ALICE*, and (c) *SheLL* on the original layout of *AES*.

E. Ablation Study

We analyze the impact of removing different feature groups, loss terms, and selection filters on *AES* and *IBEX* (Table III).

- *Node features.* Timing and physical features provide critical information for GNN learning and clustering. Removing them from the feature vector degrades clustering quality, which in turn results in inferior redaction outcomes. When the timing feature is removed, gates with worse slack values are dispersed across groups. Consequently, even with the selection mechanism (Section III-E), this leads to significant increases in delay for both benchmarks. Likewise, removing physical features causes insecure redaction for *AES* and a 5.1% increase in area for *IBEX*.

- *Training losses.* Loss terms determine node embeddings generated by the GNN. The timing loss encourages gates with sufficient slack to have similar embeddings. Delay surges for *AES* when this term is removed. The boundary loss encourages the formation of strongly intra-connected groups while limiting excessive inter-group connections. Removing this term causes significant delay overhead on both benchmarks.

- *Security constraint.* Eliminating the security constraint produces groups that are vulnerable to the SAT-based attack.

These results confirm that node features provide essential signals for clustering, loss terms enhance clustering quality, and the security constraint acts as a safeguard against the SAT-based attack.

V. CONCLUSION

This paper presents a novel approach to select subcircuits for eFPGA redaction. Using self-supervised GNN training and clustering, it partitions a circuit into groups, ensuring that nodes with similar physical locations, slack, and connectivity have similar representations. The most desirable group is then selected according to performance, cost, and security criteria. Compared to existing techniques, the proposed method consistently achieves lower timing and area overheads while remaining resilient to I/O attacks. Future work includes extending the approach to large designs and integrating it with commercial IC design flows.

REFERENCES

- [1] R. Torrance and D. James, "The state-of-the-art in IC reverse engineering," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2009, pp. 363–381.
- [2] The Commission on the Theft of American Intellectual Property, "The IP commission report," Available at: https://www.nbr.org/wp-content/uploads/pdfs/publications/IP_Commission_Report.pdf, 2013.
- [3] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of integrated circuits," in *Proceedings of the conference on Design, automation and test in Europe*, 2008, pp. 1069–1074.
- [4] Y. Li, G. Zhao, Y. He, and H. Zhou, "ObfusLock: An efficient obfuscated locking framework for circuit IP protection," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2023, pp. 1–6.
- [5] Y. Li, G. Zhao, Y. Ju, Y. He, J. Gu, and H. Zhou, "LLA: Enhancing security and privacy for generative models with logic-locked accelerators," *arXiv preprint arXiv:2512.22307*, 2025.
- [6] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 709–720.
- [7] F. Imeson, A. Emtenan, S. Garg, and M. Tripunitara, "Securing computer hardware using 3D integrated circuit (IC) technology and split manufacturing for obfuscation," in *22nd USENIX Security Symposium (USENIX Security 13)*, 2013, pp. 495–510.
- [8] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2015, pp. 137–143.
- [9] Y. Li, G. Zhao, Y. He, and H. Zhou, "DE2: SAT-based sequential logic decryption with a functional description," in *2025 Design, Automation & Test in Europe Conference (DATE)*. IEEE, 2025, pp. 1–7.
- [10] D. Sirone and P. Subramanyan, "Functional analysis attacks on logic locking," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2514–2527, 2020.
- [11] N. Limaye, S. Patnaik, and O. Sinanoglu, "Valkyrie: Vulnerability assessment tool and attack for provably-secure logic locking techniques," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 744–759, 2022.
- [12] Y. Wang, P. Chen, J. Hu, and J. Rajendran, "The cat and mouse in split manufacturing," in *Proceedings of the 53rd Annual Design Automation Conference*, 2016, pp. 1–6.
- [13] H. Li, S. Patnaik, A. Sengupta, H. Yang, J. Knechtel, B. Yu, E. F. Young, and O. Sinanoglu, "Attacking split manufacturing from a deep learning perspective," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.
- [14] Y. Li, G. Zhao, Y. He, and H. Zhou, "Evaluating the security of logic locking on deep neural networks," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.
- [15] P. Mohan, A. Atli, J. Sweeney, O. Kibar, L. Pileggi, and K. Mai, "Hardware redaction via designer-directed fine-grained eFPGA insertion," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 1186–1191.
- [16] A. Duncan, F. Rahman, A. Lukefahr, F. Farahmandi, and M. Tehranipoor, "FPGA bitstream security: a day in the life," in *2019 IEEE International Test Conference (ITC)*. IEEE, 2019, pp. 1–10.
- [17] H. M. Kamali, K. Z. Azar, F. Farahmandi, and M. Tehranipoor, "SheLL: Shrinking eFPGA fabrics for logic locking," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2023, pp. 1–6.
- [18] C. M. Tomajoli, L. Collini, J. Bhandari, A. K. T. Moosa, B. Tan, X. Tang, P.-E. Gaillardon, R. Karri, and C. Pilato, "ALICE: An automatic design flow for eFPGA redaction," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 781–786.
- [19] L. Collini, J. Bhandari, C. Muscari Tomajoli, A. Moosa, B. Tan, X. Tang, P.-E. Gaillardon, R. Karri, and C. Pilato, "Arianna: An automatic design flow for fabric customization and eFPGA redaction," *ACM Transactions on Design Automation of Electronic Systems*, 2025.
- [20] Y.-C. Lu, S. S. K. Pentapati, L. Zhu, K. Samadi, and S. K. Lim, "TP-GNN: A graph neural network framework for tier partitioning in monolithic 3D ICs," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.
- [21] Y.-C. Lu, S. Pentapati, and S. K. Lim, "VLSI placement optimization using graph neural networks," in *Proceedings of the 34th Advances in Neural Information Processing Systems (NeurIPS) Workshop on ML for Systems, Virtual*, 2020, pp. 6–12.
- [22] Z. Xie, R. Liang, X. Xu, J. Hu, Y. Duan, and Y. Chen, "Net2: A graph attention network method customized for pre-placement net length estimation," in *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, 2021, pp. 671–677.
- [23] Z. Guo, M. Liu, J. Gu, S. Zhang, D. Z. Pan, and Y. Lin, "A timing engine inspired graph neural network model for pre-routing slack prediction," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 1207–1212.
- [24] K. K.-C. Chang, C.-Y. Chiang, P.-Y. Lee, and I. H.-R. Jiang, "Timing macro modeling with graph neural networks," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 1219–1224.
- [25] Y. Liu, Z. Ju, Z. Li, M. Dong, H. Zhou, J. Wang, F. Yang, X. Zeng, and L. Shang, "Graphplanner: Floorplanning with graph neural network," *ACM Transactions on Design Automation of Electronic Systems*, vol. 28, no. 2, pp. 1–24, 2022.
- [26] L. Ferretti, A. Cini, G. Zacharopoulos, C. Alippi, and L. Pozzi, "Graph neural networks for high-level synthesis design space exploration," *ACM Transactions on Design Automation of Electronic Systems*, vol. 28, no. 2, pp. 1–20, 2022.
- [27] A. Rezaei, R. Afsharmazayejani, and J. Maynard, "Evaluating the security of eFPGA-based redaction algorithms," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, 2022, pp. 1–7.
- [28] Z. Han, M. Shayan, A. Dixit, M. Shihab, Y. Makris, and J. J. Rajendran, "FuncTeller: How well does eFPGA hide functionality?" in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 5809–5826.
- [29] T. Ajayi, D. Blaauw, T. Chan, C. Cheng, V. Chhabria, D. Choo, M. Coltella, S. Dobre, R. Dreslinski, M. Fogaça *et al.*, "OpenROAD: Toward a self-driving, open-source digital layout implementation tool chain," *Proc. GOMACTECH*, pp. 1105–1110, 2019.
- [30] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.
- [31] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.
- [32] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.
- [33] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
- [34] F. M. Bianchi, D. Grattarola, and C. Alippi, "Spectral clustering with graph neural networks for graph pooling," in *International conference on machine learning*. PMLR, 2020, pp. 874–883.
- [35] S. Lloyd, "Least squares quantization in PCM," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.
- [36] A. Rezaei, A. Hedayatipour, H. Sayadi, M. Aliasgari, and H. Zhou, "Global attack and remedy on IC-specific logic encryption," in *2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2022, pp. 145–148.
- [37] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," *arXiv preprint arXiv:1903.02428*, 2019.
- [38] Nangate Inc., "Nangate 45nm library," <http://www.nangate.com/>.
- [39] Google and SkyWater Technology, "Skywater open source pdk," 2020. [Online]. Available: <https://github.com/google/skywater-pdk>
- [40] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "ASAP7: A 7-nm finFET predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016.
- [41] X. Tang, E. Giacomini, A. Alacchi, B. Chauviere, and P.-E. Gaillardon, "OpenFPGA: An opensource framework enabling rapid prototyping of customizable FPGAs," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2019, pp. 367–374.