Genetic Algorithm-Assisted Golden-Free Standard Cell Library Extraction from SEM Images

Mengdi Zhu, Ronald Wilson, Reiner N. Dizon-Paradis, Olivia P. Dizon-Paradis,
Domenic J. Forte, Damon L. Woodard
Florida Institute for National Security, Department of Electrical and Computer Engineering, University of Florida
Gainesville, FL, USA
Email: {zhum, ronaldwilson, reinerdizon, paradiso}@ufl.edu, {dforte, dwoodard}@ece.ufl.edu

Abstract—**Reverse Engineering (RE) of Integrated Circuits (ICs) involves studying an IC to comprehend its design, structure, and functionality. This process often entails identifying the key components within the design layout, frequently utilizing scanning electron microscope (SEM) images due to their high resolution, which offers detailed views of the IC's layers. However, current approaches in IC RE generally assume access to a standard cell library for the transition from layout to netlist for functional verification, which is not always available. To overcome this limitation, we propose a golden-free automated pipeline for extracting the standard cell library from SEM layout images. Our method has achieved 100% detection rate on the AES design layout in both 90nm and 32nm technology nodes, compared to the golden reference, by relying solely on information from the contact layer. This finding highlights the potential of our approach to efficiently detect standard cells in complex layouts by focusing on the most relevant and distinctive features of the design.**

Index Terms—**Hardware Assurance, Standard Cell Extraction, Pattern Recognition, Genetic Algorithm**

I. Introduction

Reverse Engineering (RE) of Integrated Circuits (ICs) entails examining an IC to gain insights into its design, structure, and functionality. This process typically requires breaking down the IC using various techniques to reconstruct its design and operation. This analysis can be performed at various stages of the electronic design cycle, which can be performed either non-destructively or destructively. An example of destructive IC reverse engineering is IC delayering. The IC layers can be imaged using scanning electron microscopy (SEM). A key aspect of IC reverse engineering is standard cell extraction, which aims to identify the fundamental building blocks within the IC design layout, often using IC SEM images. Successfully extracting these cells can provide valuable insights into the IC's core components, aiding with netlist reconstruction, extraction of design secrets, hardware trojan detection, and counterfeit detection [1], [2].

Traditionally, standard cell extraction has been studied using supervised methods, where reverse engineers have access to golden references, *e.g.*, standard cell libraries for specific IC designs. While these techniques can achieve great performance, they are impractical in scenarios where the golden references are unavailable. For example, the standard cell library may be inaccessible and even the techbology node may be unknown, particularly in the case of a commercial-off-the-shelf (COTS) chip. Existing research has not adequately explored IC RE from this perspective. The absence of a standard cell library significantly increases the complexity, time, and human resource requirements, as engineers may need to sift through billions of patterns to manually identify standard cells and understand their functionality. This gap limits our understanding of the key factors that lead to successful IC RE and the mitigation of hardware threats. There is a need to explore golden-free methods of IC RE, which only look at the IC layout itself.

To address these limitations, we propose a golden-free automated framework to extract potential cells from IC SEM layout images, with a particular focus on the contact layer, as they do not require a full RE delayering process to access, and the consistent patterns on them facilitate easier pattern detection [3]. This framework segments the image strip-by-strip and encodes them into unique strings for faster processing. The candidate standard cells are initially identified using a genetic algorithm-inspired pattern search method. Finally, the final list of standard cells is narrowed down using a path graph-based post-processing approach. The proposed method achieves a 100% detection rate of unique cells in the AES 90nm design only using the layout's contact layer.

In this paper, we make the following contributions:

- We proposed a novel method that is the first to apply sparse optimization, Genetic Algorithm (GA), and path graphs to RE-related tasks, introducing a new encoding system that converts layout images into strings. This approach allows for easier manipulation and faster processing of the images.
- We proposed a golden-free automated pipeline for identifying potential cells in the design layout only with the information provided in the contact layer.

The remainder of this paper is structured as follows. Section II provides background information relevant to this research, including an overview of the RE process, existing work on standard cell extraction, and the motivation for applying a GA. Section III outlines the proposed pipeline, detailing the steps involved in each phase. Section IV presents the experimental setup, results, and a discussion of the findings. Lastly, Section V summarizes the key insights and suggests directions for future work.

II. Background

This section begins with a brief overview of Integrated Circuit Reverse Engineering (IC RE), followed by a discussion of existing methods for standard cell extraction and its significance in RE. Additionally, a concise introduction to Genetic Algorithms (GA) is provided, along with the rationale for applying GA to address this research problem.

A. Reverse Engineering

RE is a process that aims to extract design information from hardware intellectual property (IP), such as ICs or printed circuit boards (PCBs) [4], [5]. This process can be used to analyze an IP at various stages of the electronic design cycle in various forms, *e.g.*, RTL, netlist, GDS layout, or on-the-field disassembly. The RE process can help detect malicious insertions or verify the inner workings of a hardware IP as long as intellectual rights are upheld. However, adversaries can use the same process to recreate the IP to unfairly compete in the market, as well as bypass any security measures in the IP [5]. Reverse engineering (RE) approaches can be divided into two categories: (1) non-destructive and (2) destructive. Non-destructive RE aims to analyze intellectual property (IP) without damaging it, using methods such as visual inspection or X-ray imaging [6]. In contrast, destructive RE renders the IP unusable after design extraction and analysis, *e.g.*, SEM imaging of delayered IC and IP disassembly. While destructive RE leaves the IP non-functional, it enables a detailed and comprehensive examination of its internal structure, materials, and components, providing insights that non-destructive methods may not reveal. This in-depth analysis enhances understanding of the product's design, functionality, and potential vulnerabilities. Beyond imaging data of physical ICs, RE can also be supplemented with additional information, such as the original design (i.e., a golden reference), which can aid in detecting modifications or tampering. RE methods solely relying on the IP itself are golden-free, making it more challenging to fully comprehend the IP's inner workings. Since a golden reference is not always accessible, particularly for potential attackers, it is crucial to understand the capabilities of golden-free reverse engineering (RE) methods to enhance our understanding of how to strengthen hardware security. In terms of golden-free IC RE methods, there are only a few automated tools [7]. Thus, there is a need for a golden-free framework that can efficiently automate the IC RE process by analyzing SEM images of delayered ICs.

B. Standard Cell Extraction

Standard cell extraction refers to identifying, isolating, and characterizing standard cells within a reverse-engineered IC design. Standard cells are basic building blocks of digital ICs, consisting of predefined logic functions, including NAND gates, NOR gates, and flip-flops, potentially of multiple drive strengths and other performance characteristics (e.g., low power, high performance, etc.). Standard cell extraction has primarily been implemented using supervised methods, where reverse engineers can access the standard cell library of specific IC designs. Previous research in this area includes pattern similarity [8] and statistical analysis [9]. Beyond direct comparison of imaging patterns, graph representations have emerged as valuable tools in standard cell recognition, as shown by GRACER [10], contributing to enhanced recognition accuracy of standard cells. Furthermore, Convolutional Neural Networks (CNNs) have been employed as feature extractors to identify representative features to compare pattern similarities [11]. In addition to finding representative features for standard cells and utilizing them for recognition, deep learning (DL) networks have been employed for an end-to-end standard cell recognition pipeline. For example, in 2020, Lin et al. [12] employed a Faster Region-based Convolutional Neural Network (RCNN) network to automatically recognize and locate standard cells, resulting in a 97.83% average precision score. Hence, many of the existing works in standard cell extraction assume access to a golden reference.

Although existing standard cell extraction methods have demonstrated strong performances, access to standard cell libraries is not always assured during RE efforts, as these libraries are often proprietary or tightly controlled, making them difficult to acquire. Additionally, deep learning-based methods for standard cell extraction demand large volumes of training data, which can be challenging to obtain in practical, real-world scenarios. These approaches are also typically tailored to specific design architectures or technology nodes, limiting their flexibility and ability to generalize across various designs and fabrication technologies. Motivated by the limitations of previous approaches, Wilson et al. [3] introduced a rule-based image processing method to extract potential cells from the contact layer of an IC design without relying on a golden library. However, this method heavily depends on predefined rules, assuming an even distribution of contact columns across the layout, which may lead to under-segmentation. Addressing such segmentation issues can be challenging, as the method is based solely on image analysis.

Given these limitations, it is essential to develop automated tools that can extract potential cells from design layouts without relying on golden references. Such "golden-free" solutions can enhance the ability to analyze and reverse-engineer designs more broadly, offering greater flexibility and efficiency, even in the absence of these references. These tools can improve the scalability of RE processes across diverse technologies and design spaces, facilitating innovation and security analysis in the semiconductor domain. Without golden references for the given design layout, the research challenge turns into an optimization problem, maximizing the information captured while minimizing the number of patterns used. Given that GA can efficiently explore large, complex solution spaces by mimicking natural selection and enabling convergence toward optimal or near-optimal solutions through iterative refinement, it is well-suited for optimization problems. Therefore, we proposed using GA for this research task.

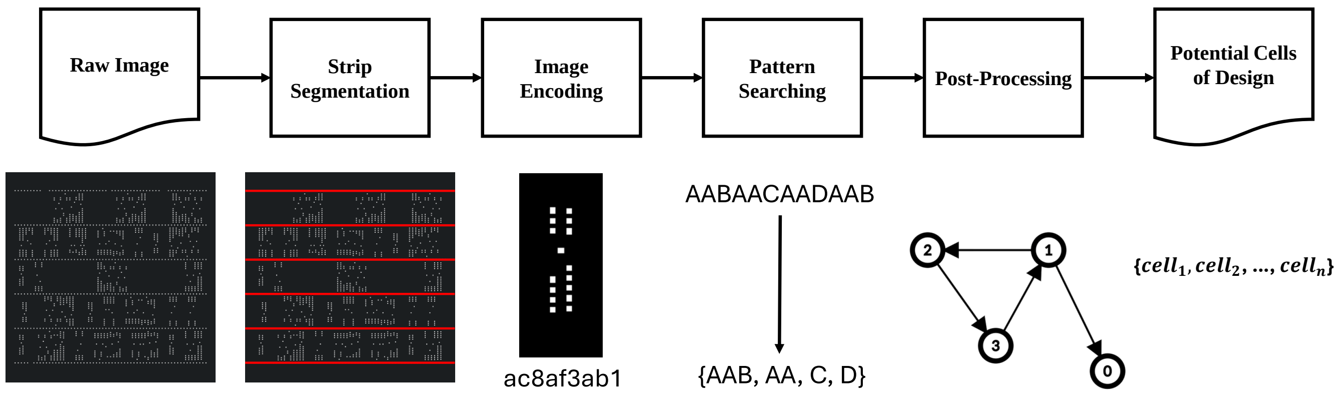


Fig. 1: Framework of proposed method

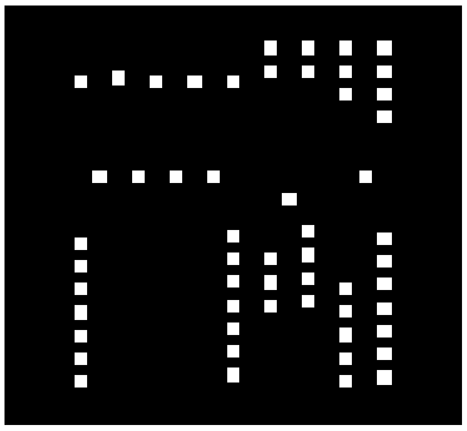
C. *Genetic Algorithm (GA)*

Genetic Algorithm (GA) is a search heuristic inspired by natural selection and genetics, used to find approximate solutions to optimization and search problems [13], [14]. It starts with a population of encoded solutions (chromosomes) evaluated by a fitness function specific to the problem. Chromosomes are selected for reproduction based on their fitness, and offspring are generated through crossover and mutation to maintain genetic diversity. This process repeats until a termination condition is met. GAs are effective for complex optimization problems and have applications in fields like operations management [15]–[17], information security [18], [19], image processing [20], wireless networking [21].

While there is currently a lack of research applying Genetic Algorithms (GA) specifically to tasks related to IC RE, existing literature indicates that GA has been successfully employed to address analogous problems in various other domains [22]–[24]. GAs present several significant advantages that make them well-suited for such applications. Firstly, they excel at exploring a wide solution space, enabling the discovery of diverse and potentially optimal solutions that traditional methods might overlook. Additionally, GAs demonstrate robustness against noise and uncertainty, making them effective in real-world scenarios where data can be imperfect or variable. Their inherent adaptability allows them to tackle complex problems where their strategies can be adjusted as needed. Furthermore, GAs can integrate domain-specific knowledge, which enhances their performance by guiding the search process based on relevant insights. Given these strengths, we chose to incorporate Genetic Algorithms (GA) into our proposed pipeline to leverage GA's capabilities and improve the efficiency and effectiveness of the overall optimization in IC RE tasks.

III. Proposed Method

Our proposed framework for golden-free standard cell extraction using genetic algorithms is shown in Figure 1. The input to our framework is the raw image of an IC's contact layer. We segment this image into strips containing a row of standard cells. These image strips are then encoded into our custom multi-step encoding scheme. We process these encoded strips using a genetic algorithm-inspired pattern search method to find candidate cells. Finally, we post-process these candidate cells to get a final set of candidates, mitigating under-segmentation and partial cell issues. The proposed framework automatically produces a set of candidate standard cells solely from the IC layout images. The processes involved in the proposed framework are elaborated in the following subsections.



(a)

ae1aa4ab6aa4ab6aa4ab6aa4ae6ad1aa0ai2ah1aa4ah7

(b)

Fig. 2: Image showing (a) NOR4X1 cell image and (b) its corresponding encoding.

A. *Strip Segmentation*

During the strip segmentation step, the layout image is divided into smaller, manageable strips, enabling a more

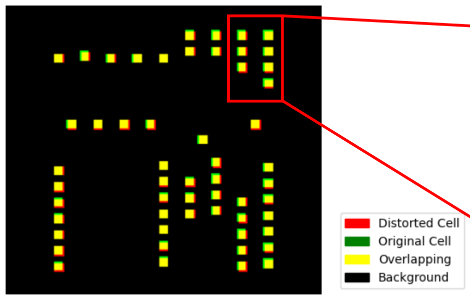


Fig. 3: Example of shape distortion with images of the same cell from cell library and design layout.

detailed and focused analysis of specific regions. The segmentation process follows the method outlined in previous work [3]. The Vcc lines in the AES design were identified by calculating the median distance between contacts in each row and marking those with the smallest median. The consistent spacing between Vcc lines was modeled using an arithmetic progression to represent their periodic nature.

B. Image Encoding

To minimize computational complexity, the 2D layout image is transformed into a 1D string through an image encoding process. This process consists of several steps, each of which is described in detail below:

- **Column Extraction:** The first step involves identifying and extracting all unique columns from the design layout. However, an additional procedure is needed to select the appropriate columns due to potential misalignment, displacement errors, and shape distortion as shown in Figure 3. This selection process involves the Hamming Distance [25] and Orthogonal Matching Pursuit (OMP) algorithm [26], which groups similar columns while choosing the most representative ones using Hamming distance to assess similarity. Algorithm 2 details this selection, where ξ indicates the percentage of information captured and Λ denotes the set of selected columns. This procedure ensures that the extracted columns accurately reflect the design layout despite minor pixel misalignments and shape distortions.
- **Column Encoding:** After selecting the unique columns, each is assigned an integer index. To maintain consistency in encoding length, these indices are converted into a specific format, such as *aa0* or *ab1*, which combines two letters and a digit to represent each column, as outlined in Algorithm 3. Columns grouped by similarity during extraction receive the same encoding. As shown in Fig. 3, a pixel shift of ±2 pixels can occur along the vertical axis due to displacement errors from floating-point operations, but this does not indicate distinct columns; rather, it reflects their high similarity. This method ensures an efficient representation of unique columns, reducing redundancy by allowing grouped columns to share a common code, resulting in a streamlined encoding scheme that accurately captures the image's structure while maintaining simplicity and efficiency.

Algorithm 1 Similarity Score

```
Require: $\Lambda$, $X$
 1: $m \leftarrow n(\Lambda)$, $n \leftarrow n(X)$
 2: Initialize $\tau$ as an empty list
 3: for i = 1 to n do
 4:     Initialize $\Phi$ as an empty list
 5:     for j = 1 to m do
 6:         $\phi = 1 - \text{HammingDistance}(\Lambda_j, X_i)$
 7:         $\Phi \leftarrow \Phi \cup \{\phi\}$        $\triangleright$ Append similarity score $\phi$
 8:     end for
 9:     $\tau \leftarrow \tau \cup \{max(\Phi)\}$
10: end for
11: return $\tau$
```

Algorithm 2 Column Selection

```
Require: set of columns $X = \{x_1, x_2, ..., x_n\}$, normalized
    column counts $y \in R^n$
 1: Initialize: $\Lambda^0 = \emptyset$, $k = 1$, $r^0 = y$, $\xi = 0$
 2: while $k \leq n$ do
 3:     $s^k = argmax(r^0)$
 4:     $\Lambda^k = \Lambda^{k-1} \cup s^k$
 5:     $r^k = y - y \cdot \text{SimilarityScore}(\Lambda^k, X)$
 6:     $\xi^k = 1 - \Sigma r^k$
 7:     $k = k + 1$
 8:     return $\Lambda^k$, $\xi^k$
 9: end while
```

Algorithm 3 Column Encoding

```
Require: Index number of Column $i$
 1: $N$ := numerical integers (0-9)
 2: $C$ := alphabet characters (a-z, A-Z)
 3: while $i > 0$ do
 4:     $e \leftarrow e + C[\lfloor i \div 100 \rfloor]$
 5:     $i = i - 100 \times \lfloor i \div 100 \rfloor$
 6:     $e \leftarrow e + C[\lfloor i \div 10 \rfloor]$
 7:     $i = i - 10 \times \lfloor i \div 100 \rfloor$
 8:     $e \leftarrow e + N[i \text{ mod } 10]$
 9:     $i = i - (i \text{ mod } 10)$
10: end while
```

- **Image Encoding:** Once the unique columns have been identified and assigned their respective encodings, the entire image is transformed into a single continuous string of encoded data. Only the relevant information, the encoded columns, is considered during this process. At the same time, any blank or empty spaces within the image are deliberately excluded to optimize the encoding efficiency. This results in a compact and precise representation of the image. Figure 2 provides examples of these image-to-encoding conversions, illustrating how each image is faithfully captured in its encoding.

C. Pattern Searching

The pattern-searching algorithm is inspired by the Genetic Algorithm (GA). The proposed pattern-searching algorithm is outlined in Algorithm 4:

Algorithm 4 Pattern-searching

```
Require: Image encoding $I$, Pop. size $N$, Stopping Criteria
1: Initialize population with candidates that are randomly
   selected sub-sequences from $I$
2: while Stopping Criteria not met do
3:     Count freq. of candidates in the current population
4:     Remove candidates with the frequency of 1
5:     Evaluate the fitness of the current population
6:     Select candidates for reproduction by frequency
7:     Mutate selected individuals to create new candidates
8:     Use the mutated candidates to form a new population
9: end while
10: return the set of candidates with the highest fitness score
```

- **Initialization:** Randomly select sub-sequences from encoding to create the initial population.
- **Fitness Function:** The fitness function, defined in Equation 2, represents the percentage of information captured by the group of candidate patterns. Additionally, Equation 1 specifies the information residue, which refers to the portion of information not captured by the set of candidate patterns.
  Given : $E_{layout}, \xi_{candidate}$
  E_{layout} := encoding of layout image
  $\xi_{candidate}$:= set of candidate patterns

$$R = E_{layout} \setminus \xi_{candidate} \tag{1}$$

$$fitness = 1 - \frac{n(R)}{n(E_{layout})} \in [0, 1] \tag{2}$$

- **Evaluate Fitness:** Assess the current population's quality by counting each individual's frequency and removing those that appear only once in the encoding. Then, the fitness score of the current population will be calculated using the fitness function.
- **Selection:** Create a parent pool for mutation by sorting the population by frequency and length of individuals and removing the individuals that are sub-strings of other individuals. The selection process also involved elitism selection, passing a certain percentage of the best individuals to next population.
- **Mutation:** Randomly select two parents from the pool and merge, split, extend, or shorten them depending on their characteristics. If the selected parents overlap, merge and split them in the overlapping section. Extend or shorten both parents by a random number of characters if they do not overlap.
- **Create New Population:** Create the next population by adapting mutated individuals and elite individuals obtained from selection.

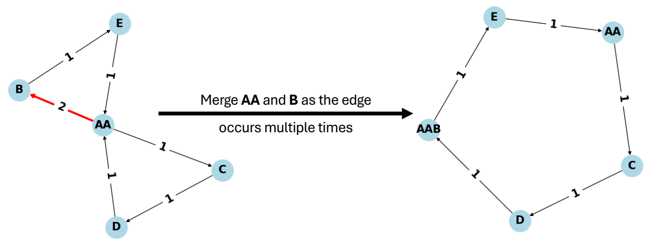


Fig. 4: Demonstration of node merge in post-processing

- **Termination:** The search process will stop if one of the following conditions is met: if the fitness score reaches the threshold or the number of iterations reaches the maximum. Once either of these criteria is satisfied, the search will be terminated.
- **Cleaning/Filtering:** The best population is filtered by removing the repeated strings to obtain the final results.

D. Post-processing

The post-processing phase is designed to resolve issues like under-segmentation and partial cells by employing a path graph-based approach. It starts by selecting potential cells from those present in the layout. Any part of the strip encoding not covered by these selected cells is broken down into individual characters. These selected cells, along with the uncovered characters, act as nodes for building the path graph.

Next, all edges in the graph are identified, extracted, and sorted by frequency. Beginning with the most frequent edge, a new path graph is constructed with an updated set of nodes if a node merge occurs. The process of extracting edges and merging nodes is then repeated. If no merge occurs, the current edge is skipped, and the process moves to the next edge.

Nodes are merged under the following conditions:

- If the nodes connected by the edge are identical, they will be merged only if the node is not identified as a sub-sequence of any existing candidate cells. If the node is recognized as a sub-sequence within an existing candidate cell, it is added to an intermediary list, which prevents any further operations on it.
- If the nodes at either end of the edge are distinct, they will be merged—provided neither node appears on the aforementioned list and the frequency of the resulting merged node is equal to or greater than the frequency of either individual node.

Figure 4 illustrates this post-processing step. The cycle continues iteratively, with new edges and graphs generated after each merge until no further updates occur in the node list. This iterative refinement helps optimize the structure of the detected cells, improving segmentation and eliminating partial cells from the cell library.

IV. Results and Discussion

This section outlines the experimental setup, details the performance of the proposed method, and provides a discussion of the findings from the experiment results.

A. Dataset

The proposed method for extracting the standard cell library is experimented on four design layouts: two DES designs and two AES designs laid out using the Synopsys 32nm and 90nm education design kit SAED[1] [27].

The AES 32nm design layout includes 41 unique cell types, with 10,185 whole cells and 103 partial cells, achieving 99.33% information capture through encoding, with discrepancies likely due to partial cells. The AES 90nm design layout, consisting of 36 unique cell types and 9,738 whole cells, achieves 100% information capture through encoding. In contrast, DES designs incorporate rotated cells that may be mirrored or inverted, leading to up to four encoding variations per cell. The DES 32nm layout has 40 unique cell types and 1,859 whole cells, while the DES 90nm layout includes 37 unique cell types and 1,815 whole cells; both achieve 100% information capture through encoding.

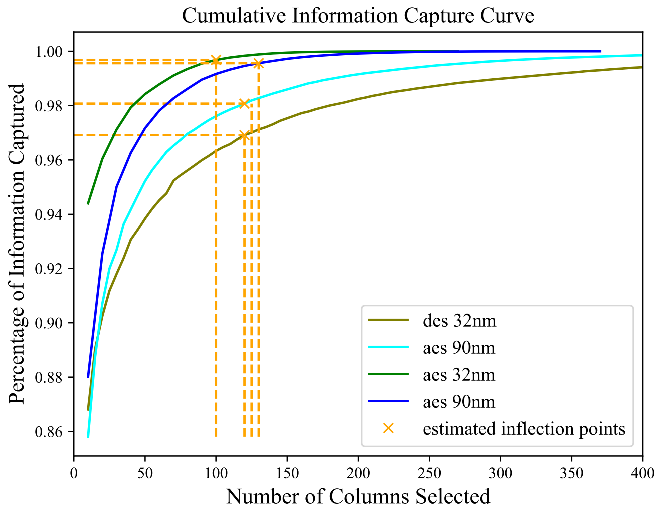


Fig. 5: Cumulative information capture curves on AES designs

B. Implementation Details

1) Strip Segmentation: The strip segmentation process divides the entire layout into narrow strips to facilitate further analysis in subsequent steps. As described in Section III, this segmentation method extracts the cell height from the layout, allowing for precise division of the layout into strips. When applied to the AES design layouts at 32nm and 90nm technologies, the extracted cell heights are 167 pixels and 288 pixels, respectively. Consequently, the 32nm layout is segmented into 107 strips, while the 90nm layout is divided into 132 strip images, enabling focused examination of specific areas in each design. Similarly, DES design layouts are segmented into 50 and 57 strips, respectively, for 32nm and 90nm node technologies.

2) Column Selection: After obtaining the layout strips, all unique columns are extracted, and a selection process is applied to encode the layout image. The column selection process aims to retain the maximum amount of information while minimizing the number of columns. Figure 5 illustrates the relationship between the amount of information captured and the number of selected columns, with the estimated inflection points corresponding to over 99% information capture rate. Consequently, the selected columns are used for the image encoding step, and any unselected columns are replaced with the most similar one from the selected set, as the similarity is evaluated with Algorithm 1. The numbers of selected columns for different designs are included in Table I

TABLE I: Summary of Parameters used in Framework

Parameters	AES 32nm	AES 90nm	DES 32nm	DES 90nm
No. of Selected Columns	100	130	120	125
Population Size N	1000	1000	1000	1000
Elitism Rate	0.10	0.10	0.10	0.10
Max Iteration	1000	1000	1000	1000
Termination Threshold	0.95	0.95	0.95	0.95

[1]https://trust-hub.org/#/data/refics

3) Pattern-Searching: Table I outlines all the parameters used in the proposed framework. The population size parameter determines how many patterns are generated in each iteration of the pattern search algorithm. The elitism rate controls the number of top-performing individuals carried over to the next generation, helping to accelerate the optimization process. The "Max Iteration" and "Termination Threshold" parameters define the stopping criteria for the pattern search algorithm. The process will end when the number of iterations reaches the maximum iteration value or when the information coverage at least the termination threshold, thereby concluding the search for patterns in the current image strip. These parameters collectively fine-tune the balance between exploration and convergence in the pattern search process, ensuring efficient and accurate detection. The same parameters are used for experiments on both AES 32nm and 90nm designs.

C. Evaluation of Proposed Framework

Table II presents a comprehensive comparison of the proposed framework's performance against previous works and the golden reference. We reassessed the baseline method from prior work [3] across all four design layouts used in this study. For the AES designs, the baseline method identified 32 and 25 unique cell types, achieving detection rates of 78.04% and 69.44%, and capturing 85.08% and 84.31% of the total information for the 32nm and 90nm node technologies, respectively. In the case of DES designs, the baseline method detected 28 and 16 unique cell types, capturing 88.31% and 32.68% of the total information for the 32nm and 90nm nodes. The lower performance on DES designs is likely due to cell rotations, which make the patterns less distinguishable in the layout images, hindering the baseline method's detection capability.

Our proposed framework accurately identified 41 and 36 unique cell types from the encoded AES 32nm and 90nm layouts, achieving 100% detection and capturing 99.33% and

TABLE II: Summary of Results

	Detection Rate				**Information Capture**				**No. of Unique Cells Detected**			
	AES 32nm	AES 90nm	DES 32nm	DES 90nm	AES 32nm	AES 90nm	DES 32nm	DES 90nm	AES 32nm	AES 90nm	DES 32nm	DES 90nm
Golden Reference	N/A	N/A	N/A	N/A	99.33%	100%	100%	100%	41	36	40	37
Proposed Method	100%	100%	87.50%	91.89%	99.33%	100%	92.96%	91.30%	41	36	35	34
Wilson et al.	78.04%	69.44%	70%	43.24%	85.08%	84.31%	88.31%	32.68%	32	25	28	16

100% of the total information. This high accuracy underscores the robustness and precision of our framework in differentiating and recognizing cell types across different node technologies. For DES designs, it detected 35 and 34 unique cell types, with detection rates of 87.5% and 91.89%, capturing 92.86% and 91.30% of the total information for 32nm and 90nm layouts, respectively.

In the DES 90nm design, the algorithm failed to detect three cell types: INVX1, INVX2, and AND4X1. An analysis of the design layout using the golden reference revealed that this issue stemmed from the assumptions of the proposed framework. Since the framework relies on pattern appearance frequencies and these three cells appeared only once in the entire design, it was unable to capture them.

For the same reason, the proposed framework also failed to detect the OR2X2 and AND2X4 cells in the DES 32nm design. Additionally, it missed three more cell types: NBUFFX2, NBUFFX4, and NBUFFX8. The failure to capture the NBUFF cells is attributed to their structural characteristics. Analyzing the contact layers of these cells revealed that NBUFF cells can be interpreted as combinations of inverter cells, as illustrated in Figure 6.

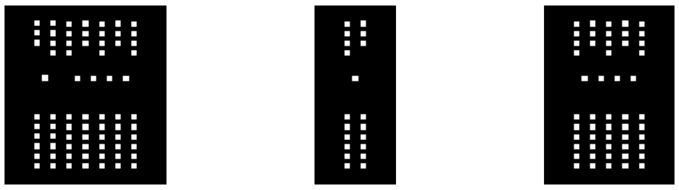


Fig. 6: Image showing (a) NBUFFX4 cell image, (b) INVX1 cell and (c) INVX4 cell.

D. Practical Considerations

While the proposed pipeline has achieved impressive performance in detecting potential cell patterns within design layouts, certain practical limitations persist that warrant further attention. One challenge arises from the stochastic nature of the Orthogonal Matching Pursuit (OMP) algorithm employed in the column selection process. Due to this inherent randomness, the algorithm may occasionally select columns with distorted shapes, which can introduce inconsistencies and variations in the resulting cell encodings. This variability could potentially impact downstream processes that rely on accurate and stable encodings.

Additionally, the column selection process prioritizes columns with higher information coverage, intentionally excluding those deemed less informative. While this strategy enhances overall efficiency, it can inadvertently lead to unique columns being assigned the same encodings as their most similar counterparts. Despite their uniqueness, these columns are grouped with others due to their perceived similarity, resulting in a loss of distinctiveness in the encoding. Ideally, such unique columns should be assigned separate encodings to preserve their individuality and ensure accurate representation in the final output. Addressing these issues could further enhance the robustness and reliability of the pipeline.

V. Conclusion and Future Work

The proposed pipeline is an automated, golden-free tool designed to extract candidate standard cells from IC design layout images. It has demonstrated the ability to achieve 100% detection rates on both AES 32nm and 90nm design layouts compared to the golden reference, relying solely on information provided by the contact layer. This approach represents a significant step forward in standard cell extraction without needing a pre-existing library or reference design.

However, variations introduced by the manufacturing process or the delayering process can lead to inconsistencies in the contact layer. For instance, the same column in the contact layer may exhibit different variations, which results in different encodings for what is essentially the same cell or column. Moreover, manual intervention or automated processes can further complicate matters, as they may introduce transformations, *e.g.*, mirroring, or rotation, causing the same cell to be encoded differently across the design.

Future work will focus on improving the quality of the design layout data to enhance detection accuracy and robustness, as well as analyzing the impact of imaging noises on the framework's performance. Additionally, future efforts plan to incorporate information from other layers of the IC design layout to further improve the overall detection performance. These improvements will strengthen the pipeline's capability to deal with the complexities and variations encountered in practical IC RE scenarios.

References

[1] C. Bao, D. Forte, and A. Srivastava, "On reverse engineering-based hardware trojan detection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 1, pp. 49–57, 2016. DOI: 10.1109/TCAD.2015.2488495.

[2] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proceedings of the IEEE*, vol. 102, pp. 1283–1295, 2014.

[3] R. Wilson, R. Y. Acharya, D. Forte, N. Asadizanjani, and D. Woodard, "A novel approach to unsupervised automated extraction of standard cell library for reverse engineering and hardware assurance," in *International Symposium for Testing and Failure Analysis*, ASM International, vol. 82747, 2019, pp. 249–255.

[4] A. T. Erozan, M. Hefenbrock, M. Beigl, J. Aghassi-Hagmann, and M. B. Tahoori, "Reverse engineering of printed electronics circuits: From imaging to netlist extraction," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 475–486, 2020. DOI: 10.1109/TIFS.2019.2922237.

[5] A. Dhavlle, *Reverse engineering of integrated circuits: Tools and techniques*, 2022. arXiv: 2208.08689 [cs.CR].

[6] N. Asadizanjani, S. Shahbazmohamadi, M. Tehranipoor, and D. Forte, "Non-destructive pcb reverse engineering using x-ray micro computed tomography," in *International Symposium for Testing and Failure Analysis*, ASM International, vol. 81030, 2015, pp. 164–172.

[7] L. Azriel, J. Speith, N. Albartus, R. Ginosar, A. Mendelson, and C. Paar, "A survey of algorithmic methods in ic reverse engineering," *Journal of Cryptographic Engineering*, vol. 11, no. 3, pp. 299–315, 2021.

[8] R. Quijada, A. Raventós, F. Tarrés, R. Durà, and S. Hidalgo, "The use of digital image processing for ic reverse engineering," in *2014 IEEE 11th International Multi-Conference on Systems, Signals & Devices (SSD14)*, 2014, pp. 1–4. DOI: 10.1109/SSD.2014.6808796.

[9] F. Courbon, "Practical partial hardware reverse engineering analysis," *Journal of Hardware and Systems Security*, vol. 4, pp. 1–10, Mar. 2020. DOI: 10.1007/s41635-019-00068-8.

[10] E. Huang, X. Hong, T. Lin, Y. Shi, and B. H. Gwee, "Gracer: Graph-based standard cell recognition in ic images for hardware assurance," *IECON 2023- 49th Annual Conference of the IEEE Industrial Electronics Society*, pp. 1–6, 2023.

[11] C. Liu, K. Wang, Q. Li, F. Zhao, K. Zhao, and H. Ma, "Novel methods for locating and matching ic cells based on standard cell libraries," *Microelectron. Eng.*, vol. 283, no. C, 2024, ISSN: 0167-9317. DOI: 10.1016/j.mee.2023.112107.

[12] T. Lin, Y. Shi, N. Shu, *et al.*, "Deep learning-based image analysis framework for hardware assurance of digital integrated circuits," in *2020 IEEE International Symposium on the Physical and Failure Analysis of Integrated Circuits (IPFA)*, 2020, pp. 1–6. DOI: 10.1109/IPFA49335.2020.9261081.

[13] T. V. Mathew, "Genetic algorithm," *Report submitted at IIT Bombay*, vol. 53, 2012.

[14] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: Past, present, and future," *Multimedia tools and applications*, vol. 80, pp. 8091–8126, 2021.

[15] S. AielloG, "Eneam," *A NonDominated Ranking Multi Objective Genetic Algorithm and ElectreMethodforUnequalAreaFacilityLayout Problems*, vol. 40, no. 12, 4812G4819, 2013.

[16] J. M. Palomo-Romero, L. Salas-Morera, and L. García-Hernández, "An island model genetic algorithm for unequal area facility layout problems," *Expert Systems with Applications*, vol. 68, pp. 151–162, 2017.

[17] C. K. H. Lee, "A review of applications of genetic algorithms in operations management," *Engineering Applications of Artificial Intelligence*, vol. 76, pp. 1–12, 2018.

[18] M. Kaur and V. Kumar, "Beta chaotic map based image encryption using genetic algorithm," *International Journal of Bifurcation and Chaos*, vol. 28, no. 11, p. 1 850 132, 2018.

[19] M. Kaur and V. Kumar, "Fourier–mellin moment-based intertwining map for image encryption," *Modern Physics Letters B*, vol. 32, no. 09, p. 1 850 115, 2018.

[20] D. Verma, V. P. Vishwakarma, and S. Dalal, "A hybrid self-constrained genetic algorithm (hsga) for digital image denoising based on psnr improvement," in *Advances in Bioinformatics, Multimedia, and Electronics Circuits and Signals: Proceedings of GUCON 2019*, Springer, 2020, pp. 135–153.

[21] B. Lorenzo and S. Glisic, "Optimal routing and traffic scheduling for multihop cellular networks using genetic algorithm," *IEEE Transactions on Mobile Computing*, vol. 12, no. 11, pp. 2274–2288, 2012.

[22] A. Adamopoulos and K. Perdikuri, "Using a genetic algorithm for detecting repetitions in biological sequences.," *WSEAS Transactions on Systems*, vol. 3, no. 4, pp. 1464–1468, 2004.

[23] A. Nasri, R. Benslimane, and A. E. Ouaazizi, "A genetic based algorithm for automatic motif detection of periodic patterns," in *2014 Tenth International Conference on Signal-Image Technology and Internet-Based Systems*, 2014, pp. 112–118. DOI: 10.1109/SITIS.2014.88.

[24] E. Heidary, H. Parvïn, S. Nejatian, *et al.*, "Automatic text summarization using genetic algorithm and repetitive patterns," *Comput. Mater. Continua*, vol. 67, no. 1, pp. 1085–1101, 2021.

[25] "Hamming distance," in *Encyclopedia of Biometrics*, S. Z. Li and A. Jain, Eds. Boston, MA: Springer US, 2009, pp. 668–668, ISBN: 978-0-387-73003-5. DOI: 10.1007/978-0-387-73003-5_956.

[26] T. T. Cai and L. Wang, "Orthogonal matching pursuit for sparse signal recovery with noise," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4680–4688, 2011. DOI: 10.1109/TIT.2011.2146090.

[27] R. Wilson, H. Lu, M. Zhu, D. Forte, and D. L. Woodard, "Refics: Assimilating data-driven paradigms into reverse engineering and hardware assurance on integrated circuits," *IEEE Access*, vol. 9, pp. 131 955–131 976, 2021. DOI: 10.1109/ACCESS.2021.3114360.