DOI 10.24412/1829-0450-fm-2025-1-80-88
УДК 621.3.049.77

Поступила: 16.01.2025г.
Сдана на рецензию: 20.01.2025г.
Подписана к печати: 22.01.2025г.

MACRO PLACEMENT OPTIMIZATION USING DIFFUSION MODELS

A. Ghazaryan

Russian-Armenian (Slavonic)University
National Polytechnic University of Armenia
gartur@synopsys.com

ABSTRACT

The component count in integrated circuits increases dramatically, which forces engineers to find new solutions for new issues. In digital methodology, placement is one of the important processes. there are many algorithms integrated in electronic design automation tools to fix placement-related issues. The most frequently faced issues are wirelength, negative slack. For solving the mentioned issues, EDA tools perform placement optimization in a multi-cycle way, which affects runtime. To fix the mentioned issue, diffusion models of machine learning have been used. With the proposed algorithm runtime has been decreased by ~15%, and the wire length has been decreased by ~6% while total negative slack has been increased by ~5%.

Keywords: Placement, diffusion models, wirelength, negative slack.

Introduction

The field of integrated circuit (IC) design has made significant progress, fostering advancements in electronics and computing. Nevertheless, the industry faces numerous challenges, with several prominent issues including design complexity, power consumption, and fabrication limitations. As the number of transistors escalates significantly, as illustrated in Figure 1, the management of design complexity emerges as a vital concern. Contemporary integrated circuits incorporate billions of transistors, necessitating the use of sophisticated design methodologies and tools [1].

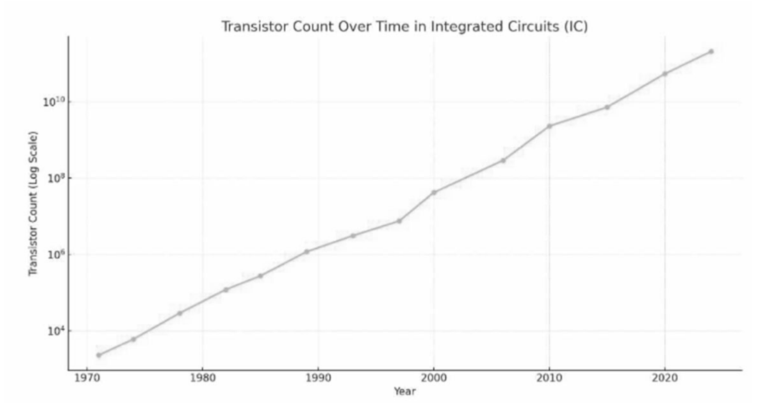


Fig 1. Transistor count over the years.

One of the methodologies used in the field is the digital methodology, also referred to as the digital design flow. This approach encompasses a variety of phases, tools, and techniques that are specifically designed to effectively convert high-level specifications into silicon [2]. A critical phase within the digital design flow is placement, which entails the strategic arrangement of circuit components on a chip layout to enhance performance, minimize power consumption, and optimize area utilization. The design incorporates macros and standard cells that must be positioned within a floorplan; an illustration of placement is provided in Figure 2.

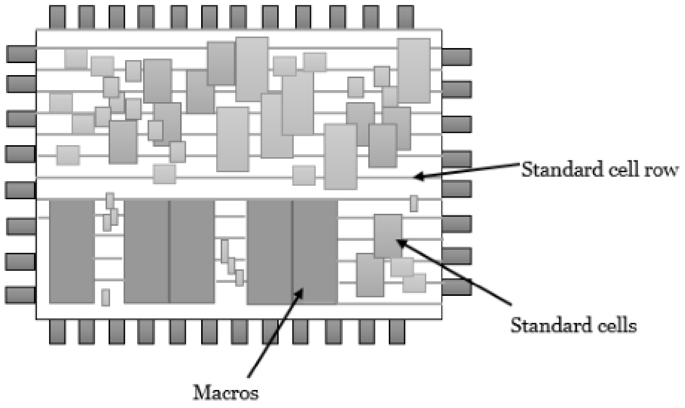


Fig. 2. Placement example.

This process addresses several challenges, including timing violations, power density, area limitations, crosstalk, and noise [3, 4].

To solve placement-related issues, many methodologies have been developed, which are used in EDA tools, some of them are:

- Analytical techniques, particularly analytical placement methods, frequently utilize mathematical models to reduce specified cost functions associated with wire length, area, and timing. The primary benefits of these analytical methods include their speed and straightforwardness, while a notable drawback is their limited flexibility [5].
- Simulated annealing is a probabilistic method. This technique enables the algorithm to avoid being trapped in local minima by permitting the acceptance of inferior solutions, with the likelihood of such acceptance diminishing over time. The primary benefits of this approach include global optimization and adaptability, while its drawbacks consist of extended execution times and sensitivity to parameters [6].
- Partitioning methods involve algorithms that segment the circuit into smaller sections, optimizing the placement within these areas to reduce interconnect lengths and improve overall performance. The benefits of this approach include scalability and decreased complexity; however, its effectiveness is contingent upon the quality of the initial partitioning [7].

Most of the electronic design automation placement tools used the above-described algorithms for the placement implementation. each of the algorithms (already stated as a classical approach) has negative and positive sides, but one negative side can be common, which is runtime, which affects on time to results. To decrease runtime and make issue issue-solving process more automated in recent years, machine learning has been widely used. Diffusion models have surfaced as a significant category of generative models within the field of machine learning. These models function by progressively converting random noise into structured data samples via a denoising process, drawing inspiration from thermodynamic concepts. Originally developed to simulate the diffusion of particles, diffusion models have become increasingly popular due to their ability to represent intricate data distributions, overcoming the constraints often associated with traditional generative adversarial networks (GANs) [8]. An overview comparing GANs and diffusion models is illustrated in Figure 3.

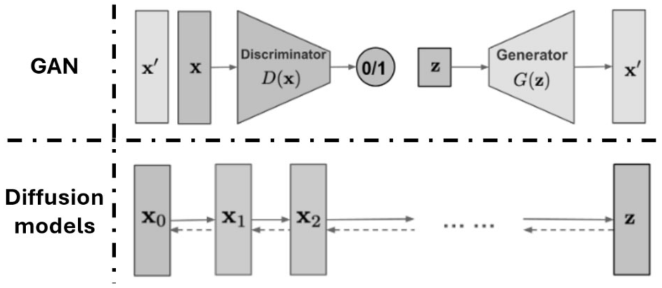


Fig 3. Overview of GAN and diffusion models.

While traditionally used in generative AI tasks, the principles behind diffusion models can be creatively adapted for macro placement optimization in VLSI design, which has been researched in the article.

Proposed method

A new method has been developed to have multi-objective optimization, optimization via wirelength and delay minimization, with a short run time. To achieve the mentioned goal, the diffusion models of machine learning have been used. The workflow can be separated in four main stages:

1. Initial placement and DEF file creation
2. Input DEF reading
3. Diffusion models-based placement optimization
4. Output DEF generation

Graphical implementation proposed method shown in Figure 4.

As mentioned, the first stage is initial placement and Design Exchange Format (DEF) file generation. The developed flow requires having input power ground mesh created NDM database, where the electronic design automation tool (in this case IC Compiler 2) can do placement, which means: initial placement without any optimization, in not optimized we will have overlaps, and big timing violations.

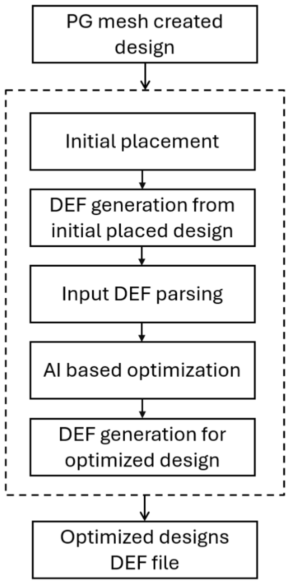

PG mesh created design
Initial placement
DEF generation from initial placed design
Input DEF parsing
AI based optimization
DEF generation for optimized design
Optimized designs DEF file

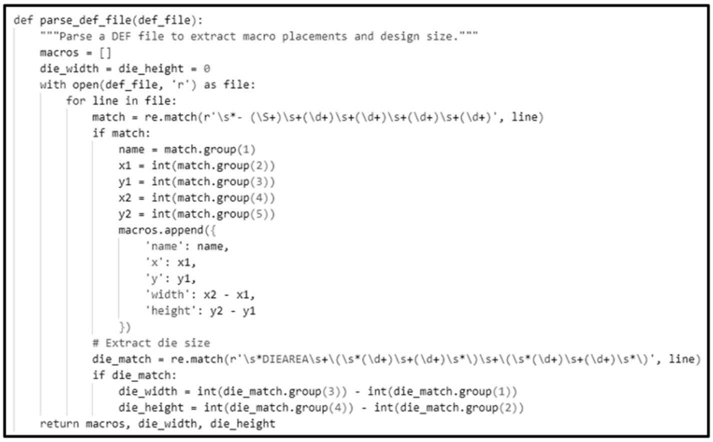
Fig. 4. Graphical approach of the proposed algorithm.

To pass the initial placed design to AI based tool, DEF file has been used. A DEF file serves as a standard format file that operate to exchange design information between different EDA tools. In the next step flow parsed the initial placed design's DEF file, which means that:

- ➢ Design size extraction
- ➢ Macro names and location extraction

DEF parser has been developed with the use of Python, the function part, and the algorithm shown in Figure 5.

While the initially placed design's DEF file is parsed, in the next stages AI optimizer starts to work. Optimization targeted to have less wirelength, less delay, and not to have overlaps and out-of-boundary cells. The optimization scripting part and algorithm are shown in Figure 6.



```python
def parse_def_file(def_file):
    """Parse a DEF file to extract macro placements and design size."""
    macros = []
    die_width = die_height = 0
    with open(def_file, 'r') as file:
        for line in file:
            match = re.match(r'\s*- (\S+)\s+(\d+)\s+(\d+)\s+(\d+)\s+(\d+)', line)
            if match:
                name = match.group(1)
                x1 = int(match.group(2))
                y1 = int(match.group(3))
                x2 = int(match.group(4))
                y2 = int(match.group(5))
                macros.append({
                    'name': name,
                    'x': x1,
                    'y': y1,
                    'width': x2 - x1,
                    'height': y2 - y1
                })
            # Extract die size
            die_match = re.match(r'\s*DIEAREA\s+\(\s*(\d+)\s+(\d+)\s*\)\s+\(\s*(\d+)\s+(\d+)\s*\)', line)
            if die_match:
                die_width = int(die_match.group(3)) - int(die_match.group(1))
                die_height = int(die_match.group(4)) - int(die_match.group(2))
    return macros, die_width, die_height
```

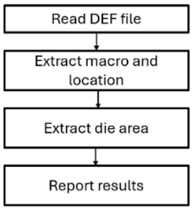
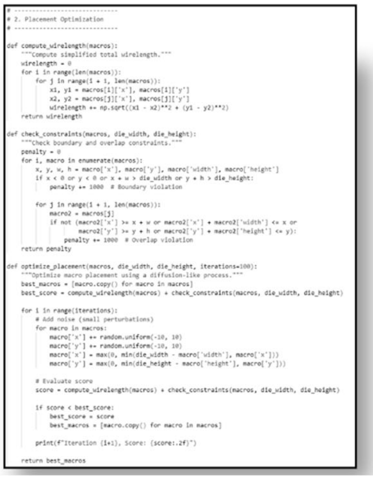


Fig. 5 DEF parser and working algorithm.



```python
# ----------------------------
# 2. Placement Optimization
# ----------------------------

def compute_wirelength(macros):
    """Compute simplified total wirelength."""
    wirelength = 0
    for i in range(len(macros)):
        for j in range(i + 1, len(macros)):
            x1, y1 = macros[i]['x'], macros[i]['y']
            x2, y2 = macros[j]['x'], macros[j]['y']
            wirelength += np.sqrt((x1 - x2)**2 + (y1 - y2)**2)
    return wirelength

def check_constraints(macros, die_width, die_height):
    """Check boundary and overlap constraints."""
    penalty = 0
    for i, macro in enumerate(macros):
        x, y, w, h = macro['x'], macro['y'], macro['width'], macro['height']
        if x < 0 or y < 0 or x + w > die_width or y + h > die_height:
            penalty += 1000  # Boundary violation

        for j in range(i + 1, len(macros)):
            macro2 = macros[j]
            if not (macro2['x'] >= x + w or macro2['x'] + macro2['width'] <= x or
                    macro2['y'] >= y + h or macro2['y'] + macro2['height'] <= y):
                penalty += 1000  # Overlap violation
    return penalty

def optimize_placement(macros, die_width, die_height, iterations=100):
    """Optimize macro placement using a diffusion-like process."""
    best_macros = [macro.copy() for macro in macros]
    best_score = compute_wirelength(macros) + check_constraints(macros, die_width, die_height)

    for i in range(iterations):
        # Add noise (small perturbations)
        for macro in macros:
            macro['x'] += random.uniform(-10, 10)
            macro['y'] += random.uniform(-10, 10)
            macro['x'] = max(0, min(die_width - macro['width'], macro['x']))
            macro['y'] = max(0, min(die_height - macro['height'], macro['y']))

        # Evaluate score
        score = compute_wirelength(macros) + check_constraints(macros, die_width, die_height)

        if score < best_score:
            best_score = score
            best_macros = [macro.copy() for macro in macros]

        print(f"Iteration {i+1}, Score: {score:.2f}")

    return best_macros
```

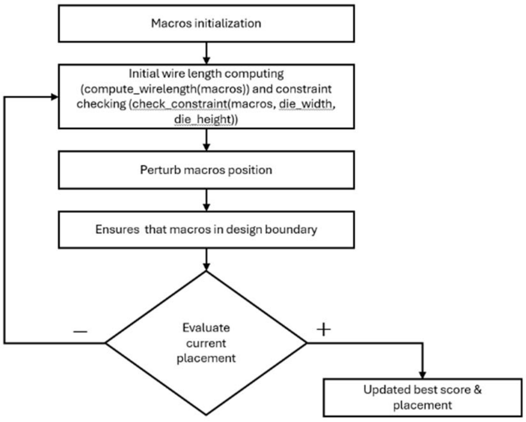
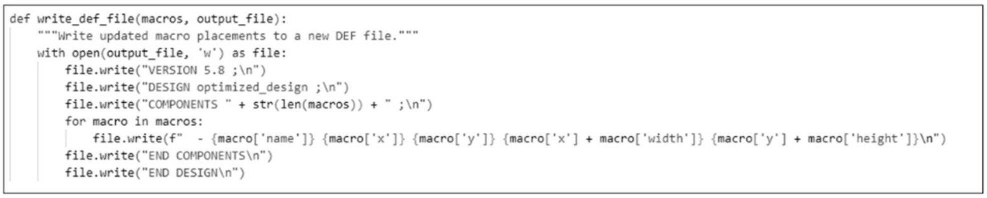

Macros initialization
Initial wire length computing (compute_wirelength(macros)) and constraint checking (check_constraint(macros, die_width, die_height))
Perturb macros position
Ensures that macros in design boundary
Evaluate current placement
−
+
Updated best score & placement

Figure 6. Optimization implementation and algorithm.

Due to optimization total wirelength for each pair has been calculated, after which the macro placement in terms of in boundary and overlap has been checked. After which, the optimize_placement function aims to optimize the placement of macros on a die area using a diffusion-like process. The goal of the function is to minimize the wire length and ensure that the macros do not overlap and that within the die boundaries.

While the optimized design has been ready, from the optimized database as an output file, DEF file is generated. The output DEF file has been generated with the write_def_file function, where optimized macros locations have been required. The write def function is shown in Figure 7.

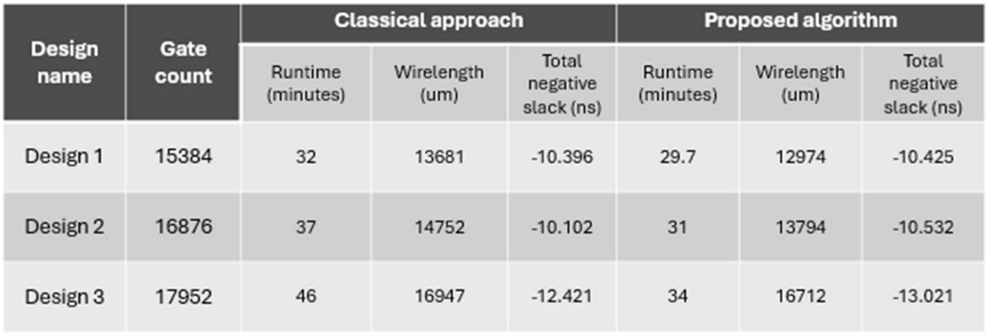


```
def write_def_file(macros, output_file):
    """Write updated macro placements to a new DEF file."""
    with open(output_file, 'w') as file:
        file.write("VERSION 5.8 ;\n")
        file.write("DESIGN optimized_design ;\n")
        file.write("COMPONENTS " + str(len(macros)) + " ;\n")
        for macro in macros:
            file.write(f"  - {macro['name']} {macro['x']} {macro['y']} {macro['x'] + macro['width']} {macro['y'] + macro['height']}\n")
        file.write("END COMPONENTS\n")
        file.write("END DESIGN\n")
```

Fig 7. Write def function implementation.

Results

To prove the proposed algorithm useless, test designs have been created; results are shown in table 1.



Design name	Gate count	Classical approach			Proposed algorithm		
		Runtime (minutes)	Wirelength (um)	Total negative slack (ns)	Runtime (minutes)	Wirelength (um)	Total negative slack (ns)
Design 1	15384	32	13681	-10.396	29.7	12974	-10.425
Design 2	16876	37	14752	-10.102	31	13794	-10.532
Design 3	17952	46	16947	-12.421	34	16712	-13.021

Conclusion

Placement is one of the most important stages in the digital design flow. During placement, engineers faced issues related to long runtimes and big wirelength. Last year's machine learning has been widely used to fix issues where humans need to act manually. To fix the above-described issues, diffusion models have been used for placement optimization. With the proposed algorithm runtime has been decreased by ~15%, and the wire length has been decreased by ~6% while the total negative slack has been increased by ~5%.

REFERENCES

1. *Tabeck P., Jain V. and Sharma A.* Technology a Sustainable Disrupter for Indian Unorganised Retail, 2022 International Mobile and Embedded Technology Conference (MECON), Noida, India, 2022, PP. 51–521, DOI: 10.1109/MECON 53876.2022.9752374.
2. *Neil H.* Weste, David M. Harris, CMOS VLSI Design, A circuits and systems perspective, fourth edition, Addison-Wesley, 2010
3. *Kahng A. et al.* Designing Low-Power Digital Systems, Wiley-IEEE Press, 2009.
4. *Brayton R. et al.* Design Automation of Digital Circuits, Kluwer Academic Publishers, 2005.
5. *Goel A. & Hu J.* (2021). Analytical Methods for Standard Cell Placement. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.
6. *Liu D. & Wong H.-S.P.* (2019). Simulated Annealing for Mixed-Size Placement. IEEE Transactions on Very Large Scale Integration (VLSI) Systems.
7. *Karypis G. & Han E.* (2020). Partitioning and Mapping of VLSI Designs. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.
8. *Ho J., Jain A., & Abbeel P.* (2020). Denoising Diffusion Probabilistic Models. *Advances* in Neural Information Processing Systems, 34.

ОПТИМИЗАЦИЯ РАЗМЕЩЕНИЯ МАКРОСОВ С ИСПОЛЬЗОВАНИЕМ МОДЕЛЕЙ ДИФФУЗИИ

А.А. Казарян

Российско-Армянский (Славянский) университет
Национальный политехнический университет Армении

АННОТАЦИЯ

Количество компонентов в интегральной схеме резко увеличивается, что заставляет инженеров искать новые решения для новых проблем. В цифровой методологии размещение является одним из важных процессов, существует множество алгоритмов, интегрированных в инструменты автоматизации электронного проектирования для устранения проблем, связанных с размещением, наиболее часто встречающимися проблемами являются длина проводов, отрицательный провис. Для решения указанных проблем инструменты EDA выполняют оптимизацию размещения многоцикловым способом, что влияет на время выполнения. Для устранения указанной проблемы были использованы диффузионные модели машинного обучения. С предлагаемым алгоритмом время выполнения было сокращено примерно на ~15%, а длина проводов была уменьшена примерно на ~6%, в то время как общий отрицательный провис был увеличен примерно на ~5%.

Ключевые слова: размещение, модели диффузии, длина провода, отрицательный провис.