INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Unlocking Modern VLSI Placement with Cutting-Edge GPU Acceleration Integrated with Deep Learning Tool Kit

Akshaya Kumar Dash[1], **Nibedita Adhikari**[2]*

1 Research Scholar, PG department of Computer Science and Application, Vani Vihar, Bhubaneswar, Odisha, India
2 Assistant Professor, PG department of Computer Science and Application, Vani Vihar, Bhubaneswar, Odisha, India

OPEN ACCESS

Received: 16-08-2024

Accepted: 27-10-2024

Published: 02-12-2024

Citation: Dash AK, Adhikari N (2024) Unlocking Modern VLSI Placement with Cutting-Edge GPU Acceleration Integrated with Deep Learning Tool Kit. Indian Journal of Science and Technology 17(44): 4600-4610. https://doi.org/10.17485/IJST/v17i44.2663

* **Corresponding author**.

nibedita.cs@utkaluniversity.ac.in

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: In VLSI, cell placement is critical in determining the overall performance, area, runtime efficiency, and power consumption of integrated circuits. The objective of this work is to find the best possible locations for the cells to meet the mentioned constraints. **Methods:** The proposed method utilizes a deep reinforcement learning strategy with heuristics to address the complexities of modern VLSI design challenges. A multi-objective deep reinforcement learning approach fused with GPU acceleration is explored to optimize placement metrics like wirelength, congestion, and runtime to gain a globally optimal placement solution. The suggested method dynamically selects appropriate parameters, a process known as Parameter Tuning, to produce high-quality placement solutions. The strategy's potency is demonstrated using open-source benchmarks sourced from storage, with BlackParrot and MemPool. **Findings:** The trial outcomes of the strategy on benchmark data show considerable improvements in placement quality. It reduces wire length by up to 4% and congestion by about 10%. Moreover, it is highly scalable and reliable in providing global placement solutions. The reduced aspect ratio indicates lower chip area utilization and reduced power consumption. **Novelty**: The integration of GPU acceleration and deep learning methodology as a strategy for VLSI global placement has not been reported in prior placement work; however, similar approaches are available for legalization and detailed placement.

Keywords: Placement; VLSI; Deep Learning; GPU Acceleration; Parameter Tuning

1 Introduction

The modern integrated circuit industry encounters major challenges due to expanding design scales and intricate objectives, resulting in extended design cycles in automated VLSI Physical Design (PD)[(1)]. The placement of circuit cells is critical in PD, and it involves macro and standard cell placement[(2)]. The Physical Design (PD) phase consists

of several manageable steps to reduce the intricacy of design. Placement plays an instrumental role in deciding the overall performance of the IC among all the steps used in fabrication. It consists of three stages: global placement (GP), legalization (LG), and detailed placement (DP). GP establishes initial cell locations, while LG and DP refine these placements to ensure compliance with design constraints[3,4]. Effective placement estimates wire length and congestion, aiding earlier design stages[5,6]. Due to the extensive numerical optimization required, contemporary placers often need several hours to complete complex designs, impeding the efficiency of design iterations. Hence, there is a consistent preference for achieving rapid and superior placement. Among all the placement methods the Analytical placement method is the preferred one and over the years its popularity has increased.

Analytical placement methods are of two types: quadratic and nonlinear approaches, are often accelerated by multithreaded CPUs. However, they rapidly saturate as the number of threads increases. GPU acceleration has been utilized in analytical placement, though it lacks validation in real operations[7]. There are techniques that improve placement quality like wire length driven cell arrangement[8] or the Hybrid Genetic-Paired-Permutation (HGPP) algorithm[9] but struggle with large designs. Similarly, the Improved Harmony Search (IHS) method[10] and multicriteria optimization techniques (MCOT)[11] show enhanced performance but struggle with scalability. The DPAHMA work of[12] and Enhanced Matrix Generation Framework (IMGF) which is a thermal-aware placement framework[13] offer improvements but require further refinement for large, intricate designs. Deep reinforcement learning (DRL) frameworks have shown significant improvements in placement parameters within commercial EDA tools, reducing the need for human intervention[14]. Hybrid algorithms like DRL, harmony search, and multicriteria optimization meet placement challenges either in terms of wirelength or congestion or runtime. Sometimes one of the criteria is met by compromising the other two. This necessitates a need for further research and optimization. Therefore, efforts have been made to address the existing challenges in the proposed methodology by leveraging GPU acceleration, converting the placement problem into a deep learning task, integrating modified essential kernels, and emphasizing kernel operators. These approaches have resulted in notable improvements in speed, reduced coding complexity, and effective congestion mitigation.

The literature study reveals the challenges in validating placement methods owing to the absence of public structures and steep costs[15]. Since the placement problem is NP-hard, deep learning and machine learning techniques are being explored to address this issue[16]. The Optimal Placement Method (Opti-PlaceM), a unique approach, combines clustering and the parallel processing capability of GPUs, to give an optimized placement solution for individual cluster. The test results show that Opti-PlaceM produces high-quality placement solutions, with the following key contributions:

1. Implementing GPU acceleration and deep learning-based methodologies substantially enhances computational speed and operational efficiency
2. The conversion of the placement problem into a deep learning training task aims to achieve precise optimization and generate placement solutions of superior quality.
3. The placement process's accuracy is enhanced by incorporating modified essential kernels for wire length and density calculations.
4. Significant improvements in rewards, cell area, power consumption, wire length, and adherence to design rules have been achieved by implementing enhanced trade-offs and reducing coding requirements.

The current work is structured into several sections of this article as outlined below. Section 2 recaps the structure of Opti-PlaceM, explaining its important attributes such as GPU acceleration, deep learning integration, and enhanced kernel operators. Section 3 compares its success with existing placement algorithms across various metrics, such as rewards, cell area, power, wire length, worst slack, design rule violations, and runtime efficiency. Section 4 describes a detailed summary of the research findings, stressing the outstanding performance of Opti-PlaceM. It also discusses challenges and suggests future improvements for VLSI placement.

2 Methodology

Modern VLSI designs, with billions of transistors and complex interconnections, overwhelm traditional placement algorithms, rendering them unscalable for large-scale circuits. As a result, these algorithms lead to inefficient placement and require significant time and computing resources. In time-sensitive sectors, prolonged design cycles hinder time-to-market.

2.1 Problem Statement

Improving runtime while simultaneously maintaining the quality of placement is the objective in almost all placement scenarios. Unsatisfactory placement can violate critical placement metrics such as signal delay, power consumption, and timing

restrictions. This drives the development of sophisticated algorithms capable of handling multiple objectives simultaneously.

Hypothesis 1: Graphics processing unit (GPU) acceleration will shorten the time it takes to calculate VLSI placement, thereby allowing faster processing of complex designs.

Hypothesis 2: Using existing placement data to train deep learning models enhances placement quality by enabling the prediction of optimal placement strategies.The placement quality can be improved by predicting the best placement scheme, which helps minimize wirelength, reduce congestion, and enhance timing performance.

Hypothesis 3: A scalable and efficient solution to current VLSI design problems may be achieved by combining GPU acceleration with deep learning-based predictions; this will be superior to standard placement methods in terms of computation time and placement quality.

2.2 Analytical Placement

The GP phase is typically the most labor-intensive component within the analytical placement. The placement algorithm seeks to reduce wire length costs while also meeting density constraints and the mathematical formulation is expressed in the following equation. The wirelength cost function, denoted as WL(), calculates the wire length of a given net instance, t. The function l() represents the layout density of a specific location while l_t represents a predetermined target density.

$$min\left(\sum_{t=0}^{T} WL(t;i,j)\right) s.tl(i,j) \leq l_t \tag{1}$$

One common strategy for solving this problem involves relaxing the limitations on density and incorporating them into the objective function as a penalty term, as shown in Equation (2).

$$min\left(\sum_{t=0}^{T} WL(t;i,j)\right) + \propto L(i,j) \tag{2}$$

The function L() represents the design's density penalty applied to disperse cells. The satisfaction of density limitations can be achieved through a gradual increase in the weight of the variable $\propto$.

2.3 Deep Learning Method Overview

An autonomous reinforcement learning (RL) agent is developed to refine the variable configurations of the placement tool. The primary goal of this agent is to minimize wire length. The RL problem at hand comprises four fundamental components:

- **State:** The states encompass the entirety of netlists worldwide, along with all conceivable combinations of variable settings (P) derived from placement tools such as Cadence Innovus or Synopsys ICC2.A singular state, denoted as s, is characterized by an exclusive netlist and a specific set of current variables.
- **Action:** Actions refer to a collection of measures an agent can employ to alter the existing variables.
- **Result of Action:** The execution of an action a results in modifying the configuration of a specific subgroup of parameters.
- **State Transition:** The state transition can be described as follows: when a state (s_x) and an action is given, the resulting state (s_{x+1}) is obtained by updating the variables of the netlist while keeping the structure unchanged.

The reward is assessed by subtracting the HPWL obtained from the placement tool. A higher reward demonstrates the action's potency in fine-tuning variable settings, especially in minimizing wire length.

The Figure 1 demonstrates the parameter tuning strategy where an agent gains knowledge in machine learning through repeated interactions with its surroundings, occurring over a sequence of steps. For each step of iteration x, it is provided with a state s_x and selects an action a_x from the available predefined action space. It is determined on the basis of its policy, denoted by π. Here, policy π is state-to-action mapping. It changes to state s_{x+1} after receiving the reward signal R_{x+i+1}. The process continues until a stopping criteria is met, and then the cycle restarts. As a matter of fact, the goal of the agent is to maximize its total cumulative reward in the future, and this can be seen from the following equation:

$$G_x = \sum_{i=0}^{\infty} \propto_i R_{x+i+1} \tag{3}$$

The symbol $\propto$ represents a factor that discounts future rewards R_{x+i+1} . An optimal policy seeks the highest possible anticipated returns or values. The value function $vf_{\pi}(s_x)$, which represents the anticipated total reward starting from the state s_x while adhering to policy π, is shown in Equation (4).

$$vf_{\pi}(s_x) = E_{\pi}(G_x|s_x) = E_{\pi}\left(\sum_{i=0}^{\infty} \propto_i R_{x+i+1}|s_x\right) \tag{4}$$

The objective is denoted as G_x, the score is denoted as S_x, and the reward is denoted as R_{x+i+1}.The expected function is designated as E_{π}.

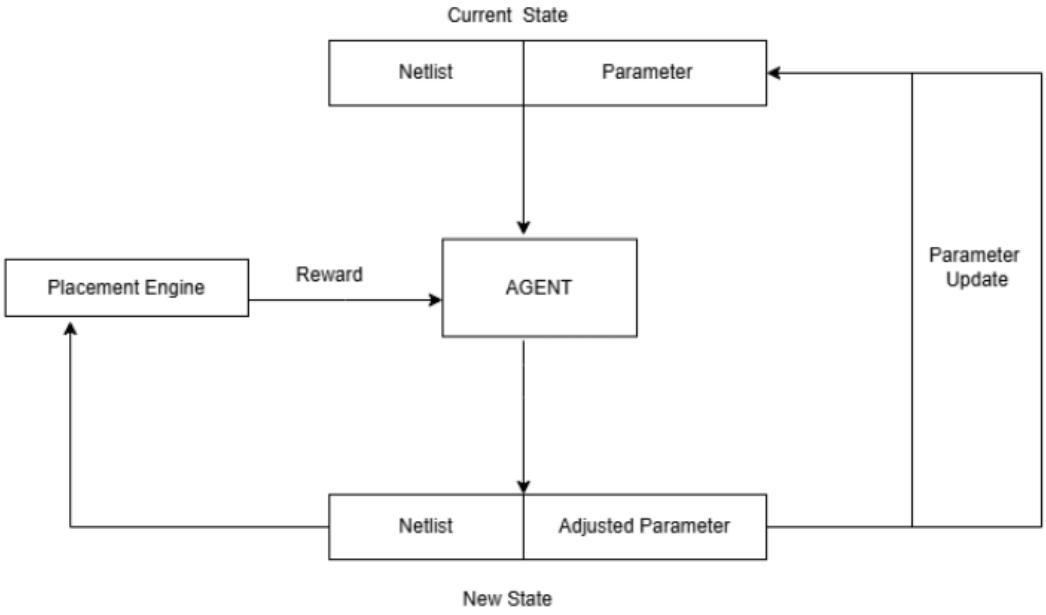

Fig 1. Parameter Tuning Through Agent Learning

2.4 Deep Learning Based Placement Strategy

The given structure incorporates both policy and value models. The Figure 2 shows the deep learning method's structure. The neural network described in this context is designed to process a state vector x composed of variable values p and netlist characteristics m. Its primary function is to generate a vector of action likelihoods, denoted as $\pi_i(x)$, for each possible action x. It also provides a scalar value $v_j(s)$ that estimates the projected accumulated reward G from a given state s.

The stratagem adjusts a placement variable setting while the value component evaluates the current location's effectiveness. The shared network structure facilitates the interaction between value and policy predictions. Gradient ascent is applied to a loss function, which includes the policy and value losses and a formalizing term. The gradient of the regularization term, value prediction, and entropy normalization are determined by policy gradients.

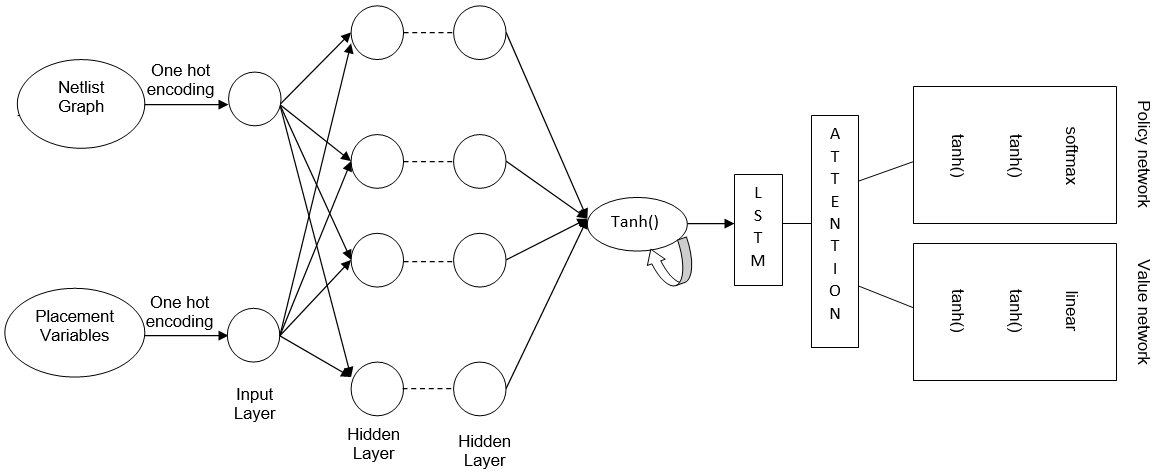

Fig 2. Deep Learning Based Placement Architecture

As shown in Figure 2, Categorical placement variables and netlist configurations are one-hot encoded and combined with extracted graph features to compute value and regulations. This input is processed through two dense layers with activations, followed by a dense linear layer. An LSTM section with layer normalization, 16 hidden units, and a forget gate is then applied[16]. Unlike feed-forward FC layers, LSTM retains information from previous inputs, enabling decision-making based on past time steps. This approach reflects the iterative nature of conventional optimization techniques. The research also incorporates a global attention system, inspired by modern NLP structures, into the Recurrent Neural Network (RNN) to prioritize important recursion components. Let h_x represent the RNN's hidden state[17], with attention agreement weights denoted as a_x corresponding to individual source hidden states, is shown in Equation (5).

$$a_x(s) = \frac{e^{(S(h_x, \overline{h}_i))}}{\sum_{i=0}^{S} e^{(S(h_x, \overline{h}_i))}} \tag{5}$$

where $S\left(h_x, \overline{h}_i\right)$ represents the alignment score between h_x and $\overline{h}_i$, and the mean is given by $\overline{h}_i$.

The alignment vector is defined using Equation (6).

$$S\left(h_x, \overline{h}_i\right) = (h_x)^T W_x \overline{h}_i \tag{6}$$

where the weight is denoted as W_x, and others as defined inEquation (5).

The alignment score indicates the weight of attention on each h_x.The attention layer then computes an aggregated output vector by uniting all the component inputs weighted by the alignment score.

The global context vector is represented in Equation (7).

$$V_x = \sum_{i=0}^{S} a_x(i)\overline{h}_i \tag{7}$$

The concealed state is paired with the attention (a_x) to generate an attentional concealed state ($\overline{h}_i$) in Equation (8).

$$\overline{h}_i = tanh(W_v(V_x \Theta h_x)) \tag{8}$$

2.5 Concurrent Autonomous Set Matching

The term "autonomous" refers to cells without interconnections, allowing the shifting cost for each cell in a standalone set to be calculated without taking into account other cells' positions. By forming a bi-colorable graph, the Linear Assignment Problem (LAP) is used to determine optimal placements within the set. It has four steps to the strategy: 1) identification and extraction of an independent set; 2) calculation of movement costs for shifting cells; 3) solving LAP; and 4) shifting cells based on the LAP solution. The process cannot be parallelized effectively because the size of independent sets is very small, mostly between 100 cells. While cost calculation of an N x N matrix can be made concurrently, the sequential extraction of autonomous sets and LAP resolution consume most of the computational resources.

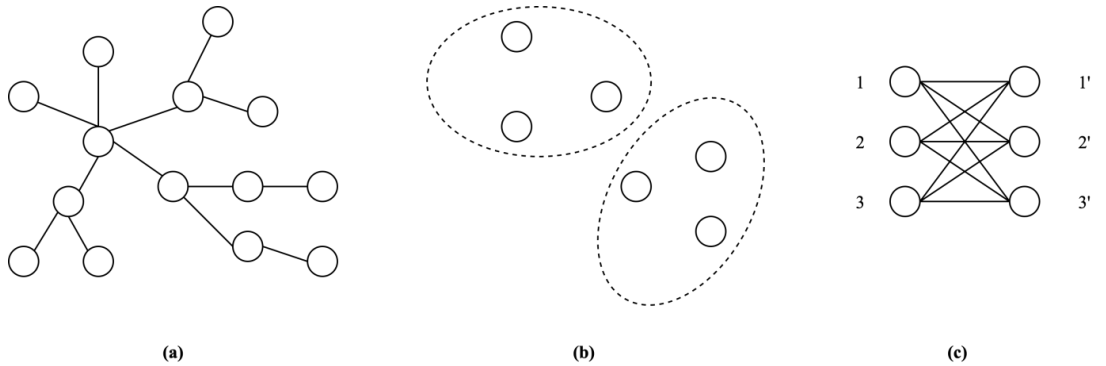

Fig 3. (a). Autonomous set (b). Partitioning (c). Linear assignment

The proposed method introduces a concurrent independent set-matching approach toward higher parallelism. Though redundant, the approach accelerates convergence due to an iteration over all cells and full parallelization of any internal process.

Figure 3 demonstrates the steps: initially extracting a large disjoint set that includes a great number of cells, and then dividing it according to the proximity and size in smaller segments. LAP instances are solved for the various subgroups, and the solution is implemented in parallel. The next section will describe the parallel execution of independent set extraction, subset division, and batch resolution of the LAP.

2.5.1 Sequential Method for Extracting Autonomous Parallel Most Excellent Sets in Graphs

The extraction of autonomous parallel sets follows a sequential procedure: if a vertex is not already in the set, it is added, and all adjacent vertices are removed. When the graph is nearly complete, with |E| approaching |V|, the complexity reduces to O(|V|). For this problem, one traversal of the graph is sufficient. Because the efficiency of the algorithm scales with the size of the graph, parallel methods might decrease the complexity of execution down to $O(\log_2|V|)$, as every vertex can be processed independently. In practice, the algorithm will terminate before the whole graph is traversed if a significant number of vertices have been gathered in order to solve the LAP in linear time, and parallelization of random order generation can be done effectively.

2.5.2 Well-Distributed K-Means Clustering

The size of the maximal autonomous set exceeds LAP method capacity, making it beneficial to divide the set into smaller subsets since most cell movement occurs locally. Cells in each subset are placed close to each other. One sequential approach divides cells into bins, using a spiral search to group nearby cells greedily. However, the runtime inefficiency of this approach because of sequential execution and erratic memory access in its GPU implementation does not cause bottlenecks for CPU implementation and thus performs well. A K-means clustering is used to locate the closest centroids using a parallel reduction from Equation (9) in order to enhance the efficiency of the GPU.

$$min\left(\sum_{x=0}^{M}\sum_{y=0}^{N}|y-\propto_x|^2\right) \tag{9}$$

The goal of K-means clustering is to find K centroids, but this can result in imbalanced clusters, unfavorable for parallel LAP solving. To address this, a weighted objective, as shown in Equation (10) is used to achieve well balanced partitioning.

$$min\left(\sum_{x=0}^{M}\sum_{y=0}^{N}W_x|y-\propto_x|^2\right) \tag{10}$$

Each group's weight, W_x, is modified during each iteration to impose a penalty on larger clusters ($\propto_x$). When a target group size N is given, the weights are initially set to 1. These weights are adjusted directly at the k_{th} iteration using the Equation below.

$$W_x(k+1) = W_x(k)\left(1+\frac{1}{2}log\left(max\left(1,\frac{|S_x|}{N}\right)\right)\right) \tag{11}$$

The target group size, N, is initially set to 1, and adjusted iteratively. When a cluster's size surpasses a threshold (N), its weight W_x increases accordingly. Experiments compute the number of groups (K) based on the ratio of nodes in the autonomous set to the group size N (set to 128). Two K-means iterations result in a well-balanced partitioning.

2.5.3 Bulk Optimization of Linear Assignment Issues

Equation (12) and Equation (13) formulated the LAP by using the Hungarian method, network flow computations, and auction methods. In the equations, Ixy =1 denotes an assignment of x to y, while Pxy denotes the weight associated with this assignment. Hungarian methods and network flow methods are quite powerful for solving sequential problems, whereas auction methods are preferred mainly for easier parallelization in distributed computing. The network simplex method and its implementation via the Lemon solver perform much faster than compared to the Hungarian method and can solve every instance of LAP on a single thread. Auction methods demonstrate 2× faster performance than the network simplex method. Auction methods involve N cells bidding for N locations, where the bid weights p_{xy} reflect the negative wire length costs. The process consists of a fully parallelizable bidding and assignment phase. While auction methods handle large N values (over 1000), in this context, LAP instances are smaller, with N around 100, and only local motions are needed. An experiment with N=128 showed that GPU execution took 5 ms, while CPU execution times were 17 ms (Hungarian), 2 ms (network simplex), and 1 ms (auction method). These findings reveal the superior efficiency of GPU execution compared to CPU.

$$max\sum_{x=0}^{M}\sum_{y=0}^{N}p_{xy}l_{xy} \tag{12}$$

$$\sum_{x=0}^{M} l_{xy} = 1 \tag{13}$$

The expansion of the variable space of Opti-PlaceM under stringent evaluation protocols and complex parameter-tuning strategies can further improve PPA metrics. The choice of placement parameters depends on design elements, such as using Nesterov's method for macro-heavy designs or Adam for more stable optimization, though not always optimal. The post-place and post-route PPA values barely coincide with those provided by the EDA tool, which demands efficient search structures in order to obtain good-quality solutions consistently. The Multi-Objective Explorer (MOE) helps identify Pareto-optimal solutions across Root Mean Square Time (RSMT), density, and congestion metrics [18,19], as shown in Figure 4. Unlike the single-objective approach used in Circuit Training, MOE enables flexible trade-offs between wire length, density, and congestion. The search process employs multi-criteria Bayesian Optimization (BO) combined with MOE to explore the large placement space and fine-tune Opti-PlaceM settings. After sampling, promising candidates are further evaluated using advanced metrics like wire length, power, and execution time in the EDA engine, following timing optimization and routing. The search generates thousands of Pareto-optimal points, and a high number of candidates due to practical constraints such as server and licensing limits and thus necessitates using k-means clustering to reduce candidate numbers. It is then assumed that points lying within close proximity in the objective solution space represent nearly identical placements, with one representative selected from each group based on the smallest RSMT. Suboptimal solutions are terminated early. This reduces the options to k = 5 candidates, providing diversified placement alternatives. The final PPA measurements are then determined by using a commercial EDA tool for these candidates, which optimally utilizes the resources while providing accurate results.

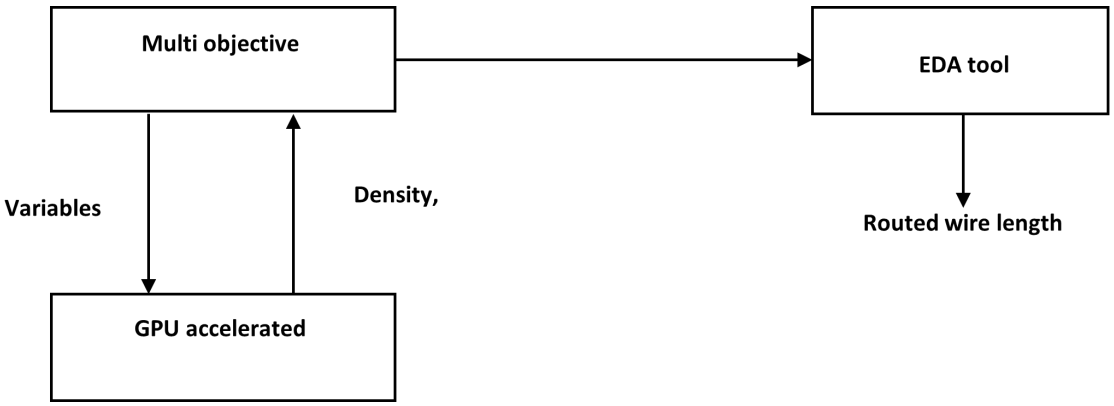

Fig 4. Computation Flow Diagram

3 Results and Discussion

The current study takes a rigorous experimental approach to assess the suggested combination of GPU acceleration and deep learning for VLSI placement, after analyzing previous research works. The objective is to prove that the proposed novel method can solve various problems like scalability, computation time, placement quality and resource utilization etc. while widening the horizon of VLSI design.

The Opti-PlaceM framework extends the ABCDPlace platforms by modifying the placement engine with Python/PyTorch and C++/CUDA [20]. MOE improvements are implemented in Python, while Tcl/Bash scripts manage the placement and routing process within the design automation tool, tightly adhering to the Macro Placement movement for open-source consistency. Deployments use Cadence Innovus 21.15 on a system with 16 Intel CPUs and 80GB memory. Register Transfer Level designs are manufactured using Cadence Genus 21.14 without Manufacturing process constraints, ensuring compatibility and reliability within the EDA environment. Open-source benchmarks, including BlackParrot and MemPool Group, are used to validate the strategy's robustness across diverse designs. The NanGate 45nm process design kit is utilized, with metrics such as reward analysis, HPWL, congestion, aspect ratio, scalability, and stability employed to assess performance. The trial results show that the proposed model outperforms top-tier techniques. Further details are discussed in the next section.

The results demonstrate promising outcomes, with significant performance improvements in the placement engine and MOE enhancements. All the results presented below compare the Opti-PlaceM method proposed in this study with several existing

placement designs, including Human, HGPP, DRL, IHS, MCOT, DPAHMA, and IMGF.

Figure 5 shows the reward percentage for each placement process iteration. Opti-PlaceM maintains average performance in rewards superior over others by 9.1% rewards, which implies a better quality of placement than those in the current models. The reason is actually that Opti-PlaceM applies the mechanism of the GPU acceleration, deepened in the formulation of placement as a task in deep learning and essential kernel formulation modified towards the calculation of wire length and density. These features accelerate global placement remarkably: the method reaches a 40-fold speedup compared with traditional methods while preserving quality similar to that achieved by advanced multithreaded placement methods.The HPWL analysis of several research investigations is depicted in Figure 6. HPWL is used to compare, and Opti-PlaceM always performs better than the other techniques with fewer iterations, yielding shorter wire lengths. It shows that Opti-PlaceM helps in reducing signal delay while improving overall circuit performance with quality that approaches advanced multithreaded placement methods.

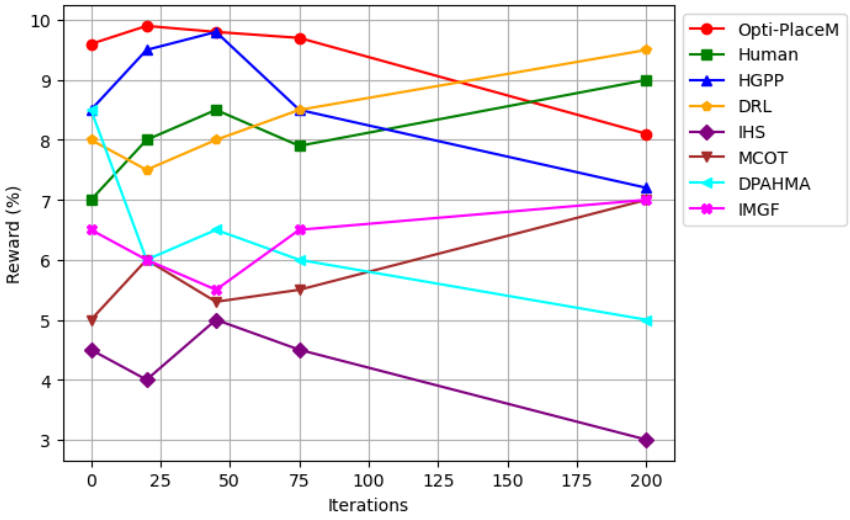

Fig 5. Rewards Analysis for VLSI Placement

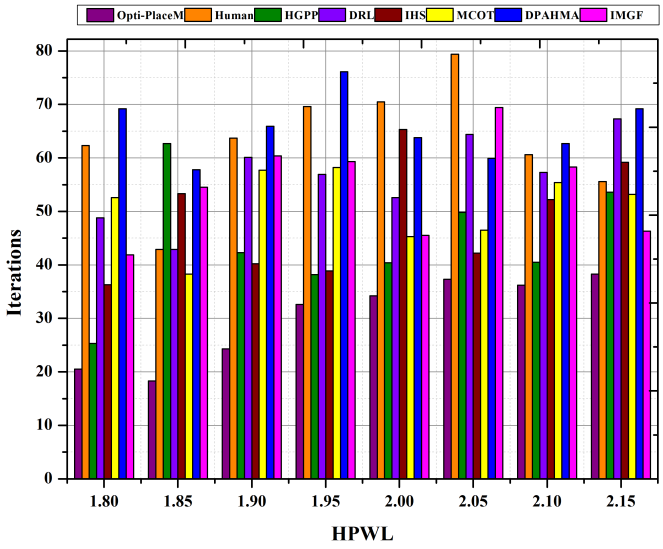

Fig 6. HPWL Analysis of the VLSI Placement

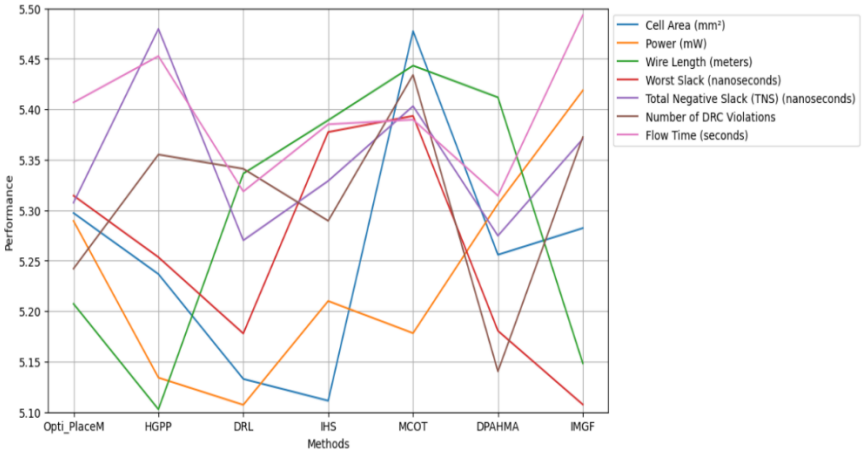

Fig 7. VLSI placement and optimization analysis

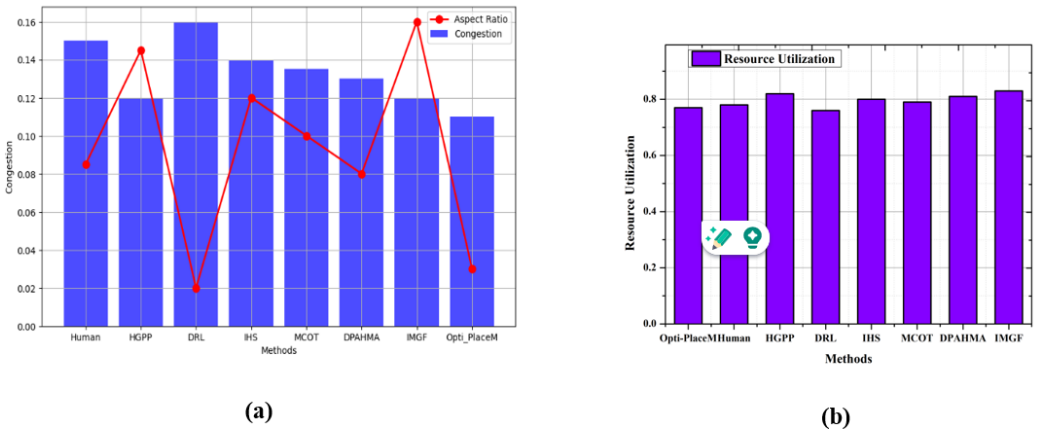

Fig 8. Analysis of (a) Congestion and Aspect Ratio, (b) Resource Utilization

Figure 7 shows performance data for many of the metrics, and cell area, power, wire length, worst slack, total negative slack (TNS), number of DRC violations, and flow time. The Opti-PlaceM is, in fact, better than alternative methods in all metrics, which means that it achieves superior trade-offs between area, power, wire length, timing, and DRC compared to existing models. Thus, Opti-PlaceM optimizes various design metrics in such an effective way that the entire performance improves from the optimized values. The Figure 8 (**a**) and (**b**) present congestion and resource utilization-based results. Opti-PlaceM presents the lowest value of congestion which is 0.1, showing lower routing congestion as compared to other methods. The aspect ratio of Opti-PlaceM is at 1.69, which is perfectly proportional. Opti-PlaceM also has an impressive resource utilization value which is 0.77, the same in comparison to the other approaches. As far as runtime is concerned, Opti-PlaceM has a value of 7.82, thus proving much more efficient over runtime as compared to alternative methods. With these Opti-PlaceM characteristics come optimized placement, reduced congestion, balanced designs, and efficient resource utilization, thereby improving runtime efficiency. The Figure 9 highlights Opti-PlaceM's superior performance in handling large, complex designs. Its high stability and scalability scores compared to others ensure reliable, consistent placement outcomes, even for demanding circuits.

All the results discussed above indicate that, compared to existing placement models, the Opti-PlaceM approach performs better on all evaluated criteria. The proposed method demonstrates a mean enhancement of around 9.1% in rewards, a decrease of approximately 5.4 mm^2 in cell area, an increase of 125mW in power consumption, a reduction of 3.46 m in wire length, an improvement of -0.2 ns in worst slack, identification of 18 design rule check violations, and a flow time of 32.18 s. This finding suggests that Opti-PlaceM offers superior overall quality and trade-offs in the placement of VLSI physical design.

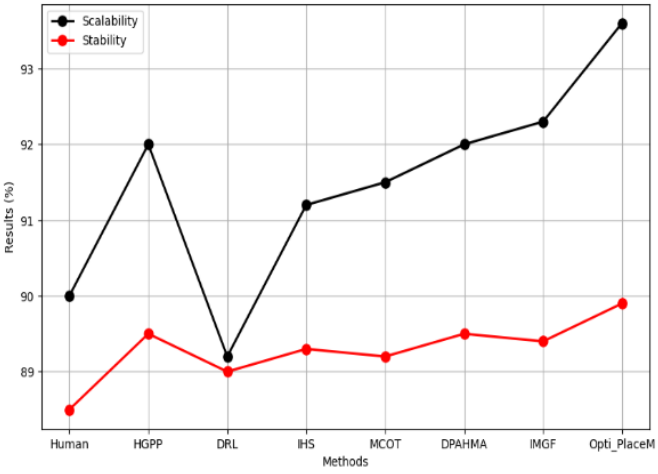

Fig 9. Scalability and Stability Analysis for VLSI Placement

4 Conclusion

An all-new VLSI placement method is introduced here in the form of Opti-PlaceM. This has leveraged GPU acceleration and deep learning toolkits to tackle the scalability, computation time, and placement quality challenges that characterize the contemporary VLSI design. The idea hypothesized here is that such an all-encompassing approach could significantly enhance the efficacy and efficiency of placement algorithms in the Electronic Design Automation sector. The prime argument here is that current state-of-the-art large-scale VLSI circuits are far beyond the reach of traditional placement methods, so new techniques ought to be there, that can handle large volumes of data and improve placement strategies.

The proposed Opti-PlaceM converts the placement problem to a training task with the use of deep learning along with GPU acceleration. The average reward at 9.1%, reduction in cell area at 5.4 mm^2, increase in power at 125 mW, 3.46 m in wire length, -0.2 ns in worst slack, 18 DRC violations, and a flow time of 32.18 s. It maintains the quality of placement similar to advanced multithreaded techniques with significantly improved runtime efficiency. Extensive experimentation shows that Opti-PlaceM outperforms state-of-the-art methods in both wire length reduction and runtime. Computational complexity, Memory Requirement, and Parallel processing are the factors that affects both Scalability and stability and the results show that Opti-PlaceM is scalable and strong across all circuit sizes and complexities, making it a realistic technology for industrial usage. Although the outcome is very promising, future work would look into the optimization of the placement by coming up with other kernel operators. A deeper integration of AI/ML can be envisioned that will automatically make placements and enhance EDA tools so as to bring in fully autonomous design systems that can dynamically adapt to constraints. That would highly increase the synergy of synthesis, placement, and routing phases. To sum up, Opti-PlaceM advances the state-of-the-art at VLSI design automation with an optimistic solution to the cell placement problem that is scalable and efficient.

References

1) Banerjee A, Basu S, Brunvand E, Mazumder P, Cleaveland R, Singh G, et al. Navigating the seismic shift of post-Moore computer systems design. *IEEE Micro*. 2021;41:162–169. Available from: https://doi.org/10.1109/MM.2021.3114899.
2) Hussain SW, Mahendra TV, Mishra S, Dandapat A. Shared matchline scheme for content addressable memory. *Integration*. 2023;88:70–79. Available from: https://doi.org/10.1016/j.vlsi.2022.08.013.
3) Lin Y, Guo Z, Mai J. Deep Learning Framework for Placement. In: Ren H, Hu J, editors. Machine Learning Applications in Electronic Design Automation. Cham. Springer International Publishing. 2022;p. 221–245. Available from: https://doi.org/10.1007/978-3-031-13074-8_9.
4) Ding H, Qian J, Huang B, Zhao L, Zhai Z. Flexible scheme for reconfiguring 2D mesh-connected VLSI subarrays under row and column rerouting. *Journal of Parallel and Distributed Computing*. 2021;151:1–12. Available from: https://doi.org/10.1016/j.jpdc.2021.01.003.
5) Lin Y, Dhar S, Li W, Ren H, Khailany B, Pan DZ. Dreamplace: Deep learning toolkit-enabled GPU acceleration for modern VLSI placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 2021;40(4):748–761. Available from: https://ieeexplore.ieee.org/document/9122053.
6) Li L, Cai Y, Zhou Q. A survey on machine learning-based routing for VLSI physical design. *Integration*. 2022;86:51–57. Available from: https://doi.org/10.1016/j.vlsi.2022.05.003.
7) Yu W, Mascagni M. GPU-Friendly FRW Algorithms for Electrostatic Computation Problems. In: Monte Carlo Methods for Partial Differential Equations With Applications to Electronic Design Automation. Singapore. Springer Nature. 2022;p. 195–216. Available from: http://dx.doi.org/10.1007/978-981-

19-3250-2_10.
8) Lin JM, Zane LC, Tsai MC, Chen YC, Lin CL, Tsai CF. PPOM: an effective post-global placement optimization methodology for better wirelength and routability. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2022;30(11):1783–1793. Available from: https://doi.org/10.1109/TVLSI.2022.3201946.
9) Ignatyev VV, Kovalev AV, Spiridonov OB, Kureychik VM, Ignatyeva AS, Safronenkova IB. Hybrid genetic-paired-permutation algorithm for improved VLSI placement. *ETRI Journal*. 2021;43(2):260–271. Available from: https://onlinelibrary.wiley.com/doi/epdf/10.4218/etrij.2019-0412.
10) Karimullah S, Vishnuvardhan D, Arif M, Gunjan VK, Shaik F, Siddiquee KN. An improved harmony search approach for block placement for VLSI design automation. *Wireless Communications and Mobile Computing*. 2022;2022(1):1–10. Available from: https://onlinelibrary.wiley.com/doi/epdf/10.1155/2022/3016709.
11) Srinivasan B, Venkatesan R, Aljafari B, Kotecha K, Indragandhi V, Vairavasundaram S. A novel multicriteria optimization technique for VLSI floorplanning based on hybridized firefly and ant colony systems. *IEEE Access*. 2023;11:14677–14692. Available from: https://ieeexplore.ieee.org/document/10042430.
12) Jiang L, Ouyang D, Zhou H, Tian N, Zhang L. DPAHMA: a novel dual-population adaptive hybrid memetic algorithm for non-slicing VLSI floorplans. *The Journal of Supercomputing* . 2023;79:15496–15534. Available from: https://link.springer.com/article/10.1007/s11227-023-05277-1.
13) Choudhury SR, Pradhan SN. An Improved Matrix Generation Framework for Thermal Aware Placement in VLSI. *IEEE Access*. 2020;8:216365–216385. Available from: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9272957.
14) Agnesina A, Chang K, Lim SK. Parameter optimization of VLSI placement through deep reinforcement learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 2022;42(4):1295–1308. Available from: https://ieeexplore.ieee.org/document/9839293.
15) Hao R, Cai Y, Zhou Q. Intelligent and kernelized placement: A survey. *Integration*. 2022;86:44–50. Available from: https://doi.org/10.1016/j.vlsi.2022.05.002.
16) Amuru D, Zahra A, Vudumula HV, Cherupally PK, Gurram SR, Ahmad A, et al. AI/ML algorithms and applications in VLSI design and technology. *Integration*. 2023;93. Available from: https://doi.org/10.1016/j.vlsi.2023.06.002.
17) Fang Z, Han J, Wang H. Deep reinforcement learning assisted reticle floorplanning with rectilinear polygon modules for multiple-project wafer. *Integration*. 2023;91:144–152. Available from: https://doi.org/10.1016/j.vlsi.2023.03.012.
18) Huang G, Hu J, He Y, Liu J, Ma M, Shen Z, et al. Machine Learning for Electronic Design Automation: A Survey. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*. 2021;26(5):1–46. Available from: https://doi.org/10.1145/3451179.
19) Attaoui Y, Chentouf M, Ismaili ZE, Mourabit AE. Machine learning application for cell delay accuracy improvement at post-placement stage: A case study for combinational cells. *Integration*. 2023;90:261–270. Available from: https://doi.org/10.1016/j.vlsi.2023.02.011.
20) Lin Y, Li W, Gu J, Ren H, Khailany B, Pan DZ. ABCDPlace: Accelerated Batch-Based Concurrent Detailed Placement on Multithreaded CPUs and GPUs. *IEEE transactions on computer-aided design of integrated circuits and systems*. 2020;39:5083–5096. Available from: https://doi.org/10.1109/TCAD.2020.2971531.