ARTIFICIAL INTELLIGENCE SYSTEM AND METHOD FOR TRANSISTOR-LEVEL PLACE AND ROUTE OF FLATTENED DIGITAL CIRCUITS

References

US PATENT DOCUMENTS

USP 11,087,060. System, method, and computer program product for the integration of machine learning predictors in an automatic placement associated with an electronic design.

USP 11,275,882. System, method, and computer program product for group and isolation prediction using machine learning and applications in analog placement and sizing.

USP 11,348,000. System, method, and computer program product for routing in an electronic design using deep learning.

USP 11,386,322. System, method, and computer program product for routing in an electronic design using deep learning.

Other References

W. T. J. Chan et al., “BEOL stack-aware routability prediction from placement using data mining techniques,” in 2016 IEEE 34th International Conference on Computer Design (ICCD). IEEE, 2016, pp. 41–48.

R. Liang et al., “DRC hotspot prediction at sub-10nm process nodes using customized convolutional network,” in Proceedings of the 2020 International Symposium on Physical Design, 2020, pp. 135–142.

H. Wang et al., "GCN-RL Circuit Designer: Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning," 2020 57th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 2020, pp. 1-6, doi: 10.1109/DAC18072.2020.9218757.

H. Liao et al., “Attention routing: track-assignment detailed routing using attention-based reinforcement learning,” in ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers Digital Collection, 2020.

H. Liao et al., “A deep reinforcement learning approach for global routing,” Journal of Mechanical Design, vol. 142, no. 6, 2020.

A. Mirhoseini et al., "A graph placement methodology for fast chip design," Nature, vol. 594, no. 7862, pp. 207–212, 2021.

Placement Optimization via PPA-Directed Graph Clustering. Yi-Chen Lu et al., MLCAD '22: Proceedings of the 2022 ACM/IEEE Workshop on Machine Learning for CAD. September 2022. Pages 1–6, 2021.

NVCell: Standard Cell Layout in Advanced Technology Nodes with Reinforcement Learning. H. Ren et al., 2021 58th ACM/IEEE Design Automation Conference (DAC), pp. 1291-1294, 2021.

Towards Machine Learning for Placement and Routing in Chip Design: a Methodological Overview. arXiv,Pub Date: 2022-02-28 , DOI:arxiv-2202.13564. J. Yan et al.,

AI/ML Algorithms and Applications in VLSI Design and Technology. D. Amuru et al., https://arxiv.org/abs/2202.10015, 2022.

Hu J. and Sachin S. Sapatnekar, A survey on multi-net global routing for integrated circuits, Integration. Volume 31, Issue 1, November 2001, pp. 1-49.

R. Luo et al., AlphaTruss : Monte Carlo Tree Search for Optimal Truss Layout Design Buildings, Volume 12, Issue 5, 641 (2022)

Silver, D., Schrittwieser, J., Simonyan, K. et al. Mastering the game of Go without human knowledge. Nature 550, 354–359 (2017). https://doi.org/10.1038/nature24270

Schrittwieser, J., Antonoglou, I., Hubert, T. et al. Mastering Atari, Go, chess and shogi by planning with a learned model. Nature 588, 604–609 (2020). https://doi.org/10.1038/s41586-020-03051-4

Field of the Invention

The present invention relates to the field of digital circuits. More particularly, the invention relates to the use of Artificial Intelligence (AI) systems and methods for an automatic transistor-level placement and routing of flattened digital circuits comprising a high number of transistors, where no domain expert information is required and beyond human capability.

Background of the Invention

Modern processors comprise blocks that perform different functions, such as input/output (I/O), memory, internet connectivity, video processing, audio processing,

arithmetic calculations, etc. These blocks are designed either by using standard cells, customized circuits, or both, in a hierarchical design.

Hierarchical design (the standard practice of processors' design) refers to an integrated circuit or part of (i.e. block/s) that is composed of multiple sub-circuits (i.e., groups of multiple standard cells) or standard cells that serve as building blocks where a global placement is carried out by positioning and interconnecting standard cells or sub-circuits. This is opposite to a flattened design where there is no hierarchy and the transistors themselves serve as building blocks. Hierarchical design is commonly practiced in order to circumvent the exponentially increasing complexity of digital circuits (i.e., with the increasing number of transistors).

The described systems and methods are in particular applicable for standard cells as well as flattened digital circuits comprising a high number of transistors that result in numerous possible placement and routing combinations, beyond that of human capability. The existing transistor-level placement and routing systems and methods are mostly limited either to a manual placement (i.e., by a human layout editor) or to automatic tools that are aimed at improving productivity, rather than PPA. The existing automatic tools use symbolic placement presets of groupings of p-type or n-type transistors of different np row patterns' alternatives (where n and p stand for n-type and p-type transistors respectively). These methods are based on pre-placed presets of commonly practiced patterns and are aimed at simplifying the layout editing task rather than optimization. They usually offer the layout editor an initial placement, for a start, and a finite set of row-placement arrangements (e.g. np, pn, nppn, pnnp, npnp, and the like) to choose from to further, manually, modify the placement to improve upon the initially proposed placement. However, this approach is limited to a relatively small number of transistors and does not use artificial intelligence methods. Furthermore, the result heavily depends on the proficiency, experience, as well as creativity, of the layout editor and it does not span the entire set of possible placements and routings.

Publicly published Artificial Intelligence methods in the context of the present invention focus on die-level optimization rather than the standard cell or circuit level. USP 11,087,060

describes a method of training a machine learning system with an existing processor's schematic diagram and its layout to predict the layout of a second processor's schematic diagram under various constraints.

USP 11,275,882 extended upon USP 11,087,060 for analog circuitry and described machine learning methods such as genetic algorithm or random forest applied to an existing processor's schematic diagram and its layout to classify (i.e. predict), based on a second processor's schematic diagram, whether two transistors of a second processor could be grouped together, sized, and oriented to automatically generate an analog layout of the second processor. However, the reported recall (86%) and precision (86%) are not adequate for actual implementation.

USP 11,348,000 and USP 11,386,322 further extended upon the former inventions by using a deep neural network for the global routing of a processor. It describes a method of training a deep neural network with an existing processor's schematic diagram and its global route data for generating detailed route data based upon, at least in part, the global route data. The deep neural network is trained using the existing processor's schematic diagram and its global route data to generate features. Then, the neural network generates a detailed global route of a second processor when it is fed with its schematic diagram.

The recent literature is abundant with research papers about using AI methods for processor design and there are already commercial products on the market (at the processor level). W. T. J. Chan et al., "BEOL stack-aware routability prediction from placement using data mining techniques" discussed, but have not disclosed, machine learning models for predicting (i.e. classification of the validity of a processor's place and route scheme) whether a processor-level placement solution is routable without conducting trials or using early global routing for the global placement and interconnecting of standard cells. The paper mainly focused on congestion and post-routing Design Rules Check (DRC - verifies as to whether a specific design meets the constraints imposed by the process technology to be used for its manufacturing) violations. Given a netlist, clock period, area utilization, aspect ratio, and Back-End-Of-Line (BEOL is the phase of the design process that focuses on the placement and routing of the various components on the chip) the number of metals and stack-specific placement, their models predicted whether a processor's placement would be routable. Their reported results for 28nm and 45nm CMOS

technologies processors were a precision of 91% and a recall of 86% which still require human intervention since there is still a significant number of false positives and few false negatives in the classification results of their test dataset.

R. Liang et al. presented at the 2020 International Symposium on Physical Design, “DRC hotspot prediction at sub-10nm process nodes using customized convolutional network”, a Convolutional Neural Network (CNN) that predicted DRC hotspots for Sub-10nm technology nodes at the early stage of routing; namely, for global routing congestion or blockage (i.e. pin access problem). They trained a modified version of a public domain segmentation CNN (U-Net) for 27 hours on twelve 7nm designs and 166 placement instances of pin images. Inference for predicting the Design Rule Check (DRC) violation hotspots in early-stage routing resulted in a 78% true positive rate (TPR) and 9% false positive rate (FPR) of pin access prediction which does not suffice for having a fully automated DRC.

Wang et al. "GCN-RL Circuit Designer: Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning", discloses a Graph Convolutional Neural Network (GCN) combined with reinforcement learning to carry out an automatic transistor sizing for porting analog circuits from one technology node to another by leveraging reinforcement learning to transfer knowledge between different technology nodes and circuit topologies for mixed-signal integrated circuits. The GCN was used to process the connection relationship between components in analog circuits. The model aimed at mixed-signal IC design so various transistor parameters as well as capacitors and resistors were used as features. However, as they reported, their model is applicable to old - planar - CMOS transistors (e.g., porting an analog design from 180nm to 45nm) and not to advanced digital, FinFET, technologies where the width of transistors is quantized to fins, for example. Furthermore, the model was limited to a fixed topology of transistors since their goal was to convert a specific analog circuit from an older technology (e.g., 250nm) to a more advanced node (e.g. 65nm). Interestingly, R. Luo et al. described a reinforcement learning algorithm for the optimization of truss layout in “AlphaTruss: Monte Carlo Tree Search for Optimal Truss Layout Design” that bears some resemblance to transistors interconnecting. A global routing method that addresses a set of nets (i.e. blocks/macros) N = {N1, N2, ..., Nk} to be routed over an acyclic-directed graph and that could be reduced to finding the shortest path

connecting two pins in the presence of wiring congestion and blockage was discussed in "A Survey on Multi-net Global Routing for Integrated Circuits" by Jiang Hu and Sachin S. Sapatnekar in Integration, volume 31, Issue 1, November 2001, pp. 1-49. They reviewed traditional approaches such as sequential routing and hierarchical routing techniques in addition to other techniques such as move-based heuristics and iterative deletion. While most traditional routing techniques focus on congestion, new objectives such as timing and crosstalk have emerged on top of congestion.

As can be seen, most of the prior art describes the use of AI systems and methods for chip-level placement and routing and is focused on the global placement of hierarchically designed digital blocks and/or standard cells over the entire, finite, die area (i.e. on a canvas) where the constraints are of macro level; i.e. cell-to-cell or block-to-block design rules or in some cases to analog circuits. Hierarchical design (the standard practice) refers to an integrated circuit or part of (i.e. block/s) that is composed of multiple sub-circuits (i.e. groups of multiple standard cells) or standard cells serving as building blocks such that the global placement is carried out by positioning and interconnecting standard cells or sub-circuits. This is opposite to a flattened design where there is no hierarchy and the transistors themselves serve as building blocks. Hierarchical design is commonly practiced in order to circumvent the exponentially increasing complexity of digital circuits with increasing number of transistors.

Commercially available tools that use AI methods optimize the chip design flow with an emphasis on improving the productivity and runtime by automating the entire design flow instead of the common manually tuned (iterative) flow. Machine-learning-driven RTL-to-GDS (Register-Transfer-Level (RTL) design defines the logical functionality of a digital design at an abstract level before specifying the circuit's physical layout. Graphic Data Stream (GDS) is a binary file that represents layout data in a hierarchical format) full-flow optimization tools that concurrently optimize the chip's floorplan in tandem with (block or cell level) routing may also improve the PPA but are implemented at the chip level; not at the transistor level.

It is therefore an object of the present invention to provide a system and method of transistors' placement and/or routing within a standard cell or a flattened digital circuit, to optimize Power, Performance, and Area (PPA).

It is another object of the present invention to provide a system and method of transistors' placement and/or routing within a standard cell or a flattened digital circuit, that is applicable for standard cells or any digital circuit comprising a large number of transistors that feature numerous placement and/or routing combinations beyond human capability; to optimize PPA.

Summary of the Invention

In an integrated circuit, a method of flattened placement and/or routing of multiple transistors using an artificial intelligence (AI) model, comprising the steps of:

a) receiving an input consisting of a circuit schematic and features;

b) performing supervised or unsupervised learning of the AI model, to produce a predicted placement and/or routing by the AI model;

c) using a cost function for optimizing transistor placement and/or routing;

d) generating an optimal placement and/or routing by:

d.1) iteratively running cycles of placement and/or routing, based on the AI model;

d.2) assigning a value to each placement and/or routing combination, according to the cost function and recording the placements and/or routings and value information across the performed iterations until meeting a predetermined metric or exceeding a predetermined number of iterations or elapsed time;

d.3) ranking each of the generated placements and/or routing according to the cost function; and

d.4) selecting the k-top (k=1,2,….) placement and/or routing combinations.

In an embodiment, the method may comprise the steps of:

a) providing a Neural Network (NN);

b) providing a Monte Carlo Tree (MCT);

c) providing a cost function;

d) receiving an input consisting of a circuit schematic, transistor properties, and a set of design rules;

e) generating a set of placement and/or routing predictions, for training the Neural Network (NN);

f) training the neural network with the generated set, to produce a predicted policy and a value;

g) generating an optimal placement and/or routing by iteratively running the Monte Carlo Tree (MCT) in cycles of M playouts being iterations of the MCT, based on the predicted policy and value generated by the neural network (NN);

h) generating an optimal placement and/or routing by:

d.1) iteratively running cycles of placement and/or routing, based on the AI model;

d.2) assigning a value to each placement and/or routing combination, according to the cost function and recording the placements and value information across the performed iterations until meeting a predetermined metric or exceeding a predetermined number of iterations or elapsed time;

d.3) ranking each of the generated placements and/or routing according to the cost function; and

d.4) selecting the k-top (k=1,2,….) placement and/or routing combinations.

In an embodiment, any sequence of nodes representing a path down to a leaf in the Monte Carlo Tree (MCT), may be a placement combination.

In an embodiment, only placement combinations of M playouts that meet the design rules may be admissible.

The cost (reward) function *L* may be:

$$L = -\left(\alpha \sum edges + \beta \sum edges\ crossings + \gamma(Total\ Area)\right)$$

where α, β, and ɣ are normalization constants, $\sum edges$ is the total length of wiring, $\sum edges\ crossings$ is the total number of wires' intersections and the *Total Area* is the total area of the placement.

Cost functions that comprise a different functional form and different metrics or any combination of area, wiring length, wiring crossings, and other metrics may be used.

The method may further comprise the steps of:

a) storing the best placement or the k-top placement (k > 1, arbitrary);
b) carrying out another cycle of M playouts and pruning the nodes that have been already visited in a preceding playout;
c) re-training the NN on the new, pruned tree with the pruned nodes, to produce a policy and a value; and
d) iteratively carrying out the process of placement and/or routing and terminating the process upon reaching a predetermined number of M playouts.

The process of optimal placement and/or routing may be terminated upon meeting a specific criterion, or after performing a predetermined number of iterations, or after a predetermined run time lapses.

An acyclic-directed graph may be used instead of a tree structure, to thereby reduce the memory size and improve speed.

The neural network may be a single neural network or comprises a combination of multiple neural networks of the same or different types.

A neural network may be selected from the group of:

- Graph Neural Networks for inputs embedding;
- Convolutional Neural Network for placement embedding;
- Dense neural network for feature embedding;
- Dense neural network for policy and value generation.

A netlist or an RTL code may serve as an input to synthesize a netlist.

The cost function may reflect the total area, the total wiring, or both, the cost function being any function that minimizes the placement area and or the wiring length.

The cost function may be any functional form of one or more of the following:

- the circuit area;
- wiring length;
- wires crossing;
- specific design rules;
- transistor properties.

In an embodiment, the method may comprise the steps of:

a) inputting the circuit netlist and its transistors' properties to a graph neural network (GNN) that generates a two-dimensional (2D) features array of size N x h, where N is the total number of transistors and h is a hidden layer size;
b) flattening the 2D array of size N x h by reducing its dimension, and feeding the flattened 2D array to MLP1, which outputs a vector of size m;
c) feeding the convolutional neural network with a vector of Unique Identification Numbers (UIN) of each transistor's position in the preceding placement and/or routing prediction and generating an output vector of size n;
d) combining the vectors of size m and n and feeding the concatenated vector into a dense NN, to generate a new policy and value.

In an embodiment, the outputs of neural networks MLP1 and CNN may be concatenated.

The output of the neural network may be a single output.

The neural network may generate a policy only; inferring a policy that maximizes the reward, or a value only to infer the value of a predicted placement.

The transistors' properties may be a unique identification number (UIN) of the position, transistor type, a threshold voltage, the width or the number of fins, a gate length or the number of fingers.

A partial list of the transistors' properties or an extended list with additional properties may be used as input to the neural network.

The input to the neural network may be a concatenated vector of the transistor UINs, a two-dimensional array of the transistors' X and Y positions, or any type of data format that contains the transistors' position information such as an image file.

The initial weights of the neural networks may be pre-trained on a smaller circuit of fewer transistors, uniformly distributed, or determined according to methods that pertain to AI.

A UIN may be a decimal or a binary number representation of a transistor position in the placement, or a one-hot encoding representation of a transistor position in the placement, wherein any unique representation may serve as a UIN.

Each type of transistor may have a variable channel width, including different numbers of fins of FinFET, or stacks of Gate-All-Around or Complementary FET.

The predetermined score may be the standard Upper Confidence Bound (UCB) score.

The embodiments herein disclose systems and methods of a flattened transistor-level placement and/or routing of digital circuits using artificial intelligence. In particular, reinforcement learning (RL) placement and/or routing of transistors within a flattened circuit is described. The RL agent interacts with the standard cell or the circuit's environment and implements the transistors' place and/or routing. An optimal transistor placement and/or routing, corresponding to a minimum value of a cost function subjected to the manufacturing design rules (DR), is generated. The embodiments herein are particularly related to the transistor-level design of standard cells and circuits. The present invention applies to any type of field-effect transistor structure such as planar CMOS, FinFET, Gate-all-Around, Complementary field-effect transistor, and the like. A FinFET is a type of FET where the conducting channel is a thin vertical fin instead of a planar CMOS

FET's surface-buried plane. A Gate-All-Around FET (GAA) is a modified transistor structure where the gate surrounds a wire-like conducting channel. A Complementary Field-Effect Transistor (CFET) – is a device type where nMOS and pMOS GAA transistors are stacked on top of each other, allowing further maximization of the effective channel width.

Unless otherwise defined, all the technical and/or scientific terms used herein have the same meaning as commonly understood by one of ordinary skills in the art to which the invention pertains. Although methods and materials similar or equivalent to those described herein can be used in the practice or testing of embodiments of the invention, exemplary methods and/or materials are described below. In case of conflict, the patent specification, including definitions will control. In addition, the materials, methods, and examples are illustrative only and are not intended to be necessarily limiting. The terms wiring, routing, and interconnecting are used interchangeably.

Brief Description of the Drawings

The above and other characteristics and advantages of the invention will be better understood through the following illustrative and non-limitative detailed description of preferred embodiments thereof, with reference to the appended drawings, wherein:

Fig. 1 is a simplified generic block diagram of a conventional implementation of a reinforcement learning system and method;

Fig. 2 is a block diagram embodiment of the present invention's implementation of a reinforcement learning system and method;

Fig. 3 is a block diagram embodiment of a neural network system of the present invention;

Fig. 4 is a representation of a circuit's netlist as an acyclic graph where A and B are inputs and Out is the output of the circuit; grey transistors are of p-type and white transistors are of n-type;

Fig. 5a illustrates a one-track transistors' type grouping design rule implementation of the present invention; grey transistors are of p-type and white transistors are of n-type;

Fig. 5b illustrates a two-track transistors' type grouping design rule implementation of the present invention; grey transistors are of p-type and white transistors are of n-type;

Fig. 6 illustrates transistors' types and threshold-voltage design rules implementation of the present invention; grey transistors are of p-type and white transistors are of n-type;

Fig. 7a-7b illustrates different design rules of transistors' fins and filler cells (designated 1 grid) embodiment of the present invention;

Fig. 7c illustrates a transistors' fins with no filler design rule implementation of the present invention; and

Fig. 7d illustrates a transistors' placement design rule following a filler cell implementation of the present invention.

Detailed Description of the Present Invention

The present invention describes a system and method for a flattened, transistors-level, placement and/or routing of standard cells and digital circuits using Artificial Intelligence (AI) methods, carried out by AI models that perform supervised or unsupervised learning. The system and method described in the present invention result in multiple possible place and/or route solutions subject to DR constraints, ranked by a cost function, that a human circuit layout editor can choose from, based on different, metrical considerations.

The systems and methods described herein relate to the use of Artificial Intelligence (AI) methods for the optimization of standard cells and flattened digital circuits at the transistor level and in particular to reinforcement learning (RL) for transistors' placement and/or routing within a standard cell or a flattened digital circuit to optimize Power, Performance, and Area (PPA).

The proposed system and method present (with or without a canvas constraint) a local placement and/or routing of transistors, within standard cells and flattened digital circuits. The presented flattened placement and/or routing methods of transistors are subjected to local, intra-cell/circuit, design rules constraints such as the minimal distance between transistors of different threshold voltages, abutment constraints, and the like (there are a few thousands design rules that must be met in modern CMOS technology nodes). The embodiments herein describe a local routing method that addresses a set of vertices V (i.e., transistors) where V = {V1, V2, ..., Vk} to be routed over an acyclic-directed graph to find the minimal sum of the total length of (edges) connections of all pins, the minimal area subjected to design rules constraints or any other metric or metrics combination. The method of the presented invention enables optimal placement and/or routing of high

transistor-count flattened circuits (that is beyond human capability due to the numerous possible placement and/or combinations) that significantly saves labor and time and outperforms human capability. The described invention is particularly effective for a large design (i.e., of many transistors) and a large number of design rules in the presence of complex trade-offs. Furthermore, the present invention allows for flattening a complex hierarchical structure of a digital integrated circuit that comprises numerous standard cells and circuits into a single layer of transistors that will undergo placement and/or routing as a single circuit; to improve PPA. The proposed method applies to any type of field-effect transistor (FET) structure such as planar CMOS, FinFET (a type of FET where the conducting channel is a thin vertical fin instead of a surface-buried plane), Gate-All-Around FET (GAA, a modified transistor structure where the gate surrounds a wire-like conducting channel), Complementary Field-Effect Transistor (CFET, an architecture where nmos and pmos GAA transistors are stacked on top of each other, allowing further maximization of the effective channel width), as well as any future FET architecture.

Fig. 1 presents a generic block diagram of a Reinforcement Learning system for an unsupervised learning placement and/or routing for saving area and/or power consumption or improving any other metric or metrics combination. The placement determines where to place transistors in a generally limited amount of space. This is followed by routing, that determines how to connect the placed transistors; no domain expert information or data set is required. First, circuit information inputs 1 are fed to a Monte Carlo Tree 2 (MCT); MCT search (MCTS) is a heuristic search algorithm that allows for searching the best moves to "play" by relying on statistics rather than full exploration (i.e., instead of systematically checking all possible plays and check each move) as was described by Silver, D., Schrittwieser, J., Simonyan, K. *et al.* in "Mastering the game of Go without human knowledge", Nature 550, 354–359 (2017). It allows for efficiently navigating a large state space in comparison to an exhaustive search.

In an embodiment of the present invention MCTS is implemented for searching of potential admissible transistors' placements subject to design rules constraints. A Neural Network 3 is trained to learn the optimal placement and/or routing, based on intermediate results that are generated by the MCT. The training stage uses placement and/or routing data from

previous MCT runs (i.e., simulations) to predict a policy (i.e., the probability distribution of potential actions in given states) and a value (i.e., the expected reward of choosing a sequence of states). A state is a unique configuration of transistors' positions in a 2D space (x-position and y-position). Once trained, neural network 3 is integrated into a new MCT run in an iterative process. Neural network 3 is also used to evaluate the states by providing policy and value estimates which are used for the tree search (to select more promising paths of the search space). In an embodiment of the present invention, an Upper Confidence Bound (UCB) formula (that uses uncertainty in the action-value estimates for balancing between exploration and exploitation) is used to select actions. The UCB formula balances exploitation (i.e., pick the best known action) and exploration (i.e. explore new actions).

In another embodiment of the present invention, a Neural Network (NN) is used to select actions.

Yet in another embodiment, other algorithms could be implemented for action selection. The placement and/or routing generation process progresses iteratively, recording and evaluating placements and/or routings and value information across the performed iterations, until it concludes (for example, upon meeting a predetermined metric or exceeding a predetermined number of iterations or elapsed time), resulting in a ranked set of optimized transistor placements and/or routings.

The neural network may be a Graph Neural Network for inputs embedding (described for example, in Khemani, B., Patil, S., Kotecha, K. et al. A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. J Big Data 11, 18 (2024); a Convolutional Neural Network for placement embedding (described for example, in Zhao, X., Wang, L., Zhang, Y. et al. A review of convolutional neural networks in computer vision. Artificial Intelligence Review 57, 99 (2024); Dense neural network for feature embedding (described for example, in Leonardo F.S. Scabini and Odemir M. Bruno, Structure and performance of fully connected neural networks: Emerging complex network properties, Physica A: Statistical Mechanics and its Applications, Volume 615, (2023); or Dense neural network for policy and value generation.

In another embodiment of the present invention a neural or neural networks are implemented to learn the dynamics of the "game" itself (i.e., a neural network for approximating the dynamics of the environment) to estimate how good a move is as well as the result of taking it. The neural network is set to represent the dynamics of the environment with no training or constraints as was described by Schrittwieser, J., Antonoglou, I., Hubert, T. *et al.* Mastering Atari, Go, chess and shogi by planning with a learned model. Nature 588, 604–609 (2020). https://doi.org/10.1038/s41586-020-03051-4.

For a circuit comprising N transistors, the depth of the MCT tree is N, i.e., the total number of transistors. Each sequence of child nodes. i.e., a path, of the tree, 2, (down to a leaf) represents a possible placement. For n ≤ N leaves there can be 2^n placement combinations. In an embodiment of the present invention, Reinforcement Learning is used either hierarchically in several steps or in a single step comprising an environment, tasks, agents, and experiments for the placement and/or routing the plurality of transistors.

Fig. 2 presents an implementation of the block diagram of Fig. 1, according to an embodiment of the present invention. A circuit schematic, transistor properties, and a set of design rules serve as input 6. The output 13 is the best placement and/or routing or a group of top-ranked transistors' placements (i.e., transistors positions and interconnections on a semiconductor wafer) and/or their interconnecting wires. In an embodiment of the present invention, a netlist is used as input (a netlist a is an abstract representation of a digital circuit). In another embodiment, an RTL code (Register-Transfer Level - a design abstraction which models a synchronous digital circuit in terms of the flow of digital signals (data) between hardware registers, and the logical operations performed on those signals) serves as an input to synthesize a netlist. Yet in another embodiment, any data file that contains a schematic and features of a digital circuit may be used (e.g., an image file). The cost function (i.e., reward, 9) is the total circuit area, the total wiring, or both. Yet in another embodiment, the cost function is any functional form of the circuit area, wiring length, wire crossings as well as specific design rules and transistor properties.
An initialization process comprising M playouts (i.e., M iterations of the MCT) is used to create a data set for training the neural network 12. The neural network 12 is trained on the generated data set to produce policy and value, 11.

In the sequel, the MCT 7, is iteratively run (in cycles of M playouts), based on the neural network (12) predicted policy and value, to generate an optimal placement and/or routing. Any sequence of nodes (a path) down to a leaf is a placement combination. Of all placement combinations of M playouts only those that meet the design rules are admissible. The AI model is configured to provide predicted placement and/or routing.

Each of the generated placements is screened and selected according to the design rules' constraints, 8, and assigned a value (a reward) according to the cost function, 9.
In one embodiment. The cost (i.e. reward) function, 9, is:

$$L = -(\alpha \sum edges + \beta \sum edges\ crossings + \gamma(Total\ Area)) \quad \text{[Eq. 1]}$$

Where α, β, and ɣ are normalization constants, $\sum edges$ is the total length of wirings, $\sum edges\ crossings$ is the total number of wires' intersections and Total Area is the total area of the placement. In another embodiment of the present invention, any function that minimizes the placement area and/or the wiring length can be used. Yet in another embodiment, other cost functions comprising a different functional form and different metrics or any combination of area, wiring length, wiring crossings, and other metrics can be used.

The system records the placements and or/routings and the reward information across the M playouts. For an MCT's (7) of depth d < N, d child nodes are repeatedly selected using the standard Upper Confidence Bound (UCB) score or some other score until a leaf node is reached and a placement is generated. In another embodiment of the present invention, a genetic algorithm or any other combinatorial optimization selection algorithm is used to select child nodes down to a leaf. The best placement or the k-top (ranked) placements (k > 1, for example, k=1,2,3,….) are stored and another cycle of M playouts is carried out after pruning the nodes that have been already visited in a preceding playout such to reduce runtime. Neural Network 12 is re-trained on the new pruned tree to produce policy and value, 11. The process is carried out iteratively, until termination (e.g., until meeting a predetermined metric or exceeding a predetermined number of iterations or elapsed time), where the k-top (k=1,2,….) placement and/or routing combinations are selected. In an embodiment of the present invention, the process is terminated upon

reaching a pre-determined number of M playouts. In another embodiment of the present invention, the process is terminated based on a metric that compares the best placement result against the second-best placement result. Yet in another embodiment, the process is terminated upon meeting a specific criterion, or after performing a predetermined number of iterations, or after a predetermined run time lapses. In another embodiment, the process is terminated after a predetermined number of iterations or after a predetermined elapsed time.

In an embodiment of the present invention, an acyclic-directed graph is used instead of a tree structure 7 to reduce memory size and improve runtime.

In an embodiment of the present invention, the neural network (12) comprises a single or multiple neural networks of the same or different types.

Fig. 3 presents a block diagram of the neural network 12, according to one embodiment of the present invention.

A circuit netlist, 15, and its transistors' properties, 14, are inputted to a Graph Neural Network (GNN), 17, that generates a two-dimensional (2D) features array of size N x h where N is the total number of transistors and h is a hidden layer size (i.e., h is a hyperparameter of the GNN).

The 2D array of size N x h is flattened and fed to MLP1, 18, which outputs a vector of size m. The convolutional neural network, 19, is fed with a vector of Unique Identification Numbers (UIN) of each transistor's position and generates an output vector of size n. The netlist, 15, is an acyclic-directed graph comprising N vertices (i.e., N transistors). Each transistor has 3 pins (Gate, Source, Drain), and each two pins of different transistors are interconnected. Therefore, the edges are described by an array of size [3N, 2] that is fed to the GNN, 17.

The outputs of neural networks MLP1, 18, and CNN, 19, are combined and fed into MLP2, 20, the output of which is the policy and value. In an embodiment of the present invention, the outputs of neural networks 19 and 18 are concatenated. In another embodiment of the present invention other functions of combining the outputs of neural networks 19 and 18 can be applied.

In an embodiment of the present invention, the output of neural network 20 is a single output. In an embodiment of a single output, the neural network 20 generates only a

policy, inferring a policy that maximizes the reward; or generates only a value to infer the value of a predicted placement/and or routing.

In an embodiment of the present invention, the transistors' features 14 are a unique identification number of the x-position and y-position, transistor type, threshold voltage, the width or the number of fins, a gate length or the number of fingers.
In another embodiment of the present invention, a partial list of the above transistors' properties or an extended list with additional properties is used as input to neural network 17.
In an embodiment of the present invention, the input to the neural network 19 is a concatenated vector of the transistor UINs. The concatenated vector may be fed into a dense NN, to generate a new policy and value.

In another embodiment, the input to the neural network 19 is a two-dimensional array of the transistors' positions (i.e. X-position and Y-position). Yet, in another embodiment, the input to the neural network 19 is any type of data format that contains the transistors' position information such as an image file, etc.

In an embodiment of the present invention, the initial weights of the neural networks 17-20 are pre-trained on a smaller circuit (i.e., of fewer transistors), uniformly distributed, or determined according to methods known to one of ordinary skill in the art that pertains to AI (e.g., random, normal distribution, and the like) or based on domain expert information.

In an embodiment of the present invention, a UIN is a binary number representation of a transistor position in the placement (e.g., transistor number 5, 8, 24, 101, etc.). In another embodiment, a UIN a one-hot encoding as described in Klimo, M., Lukáč, P. and Tarábek, P. "Deep Neural Networks Classification via Binary Error-Detecting Output Codes" in Applied Sciences, 11(8), (2021) (https://doi.org/10.3390/app11083563) is used as a representation of a transistor position in the placement. Yet in another embodiment of the present invention, any unique representation may serve as a UIN.

In an embodiment of the present invention MLP1 (18) is a dense neural network (i.e., a fully connected neural network), MLP2 (20) is a Dense Neural Network and CNN (19) is a convolutional neural network. Yet in another embodiment of the present invention, any known type of a neural network is implemented for neural networks 17-20.

Fig. 4 is an illustration of a circuit comprising six transistors of two types. Each transistor is represented by a rectangle. Grey transistors 24-26 are of p-type and white transistors 27-29 are of n-type. Each transistor has 3 pins (Gate, Source, Drain). For clarity, only the Gate pin is denoted. The inputs and output of the circuit are A (21) and B (22) and output (23) respectively.

In an embodiment of the present invention, transistors of p-type only, n-type only, or both types comprise a circuit.

In an embodiment of the present invention, the number of transistors is unlimited (subjected to contemporary memory limitation). In an embodiment of the present invention, a circuit may comprise a single type of transistor or two types of transistors of equal or different numbers. In an embodiment of the present invention, a circuit may comprise multiple inputs and outputs, i.e. two or more; inputs or outputs.

In another embodiment, a transistor is a dummy transistor that is not functionally active or the terminals of a transistor are connected to act as a pseudo-diode, a resistor, or a capacitor.

Fig. 5a presents an embodiment of a one-track, in-track, same-type transistor placement design rules. Grey rectangles 30-32 and rows 37-38 are p-type transistors and white rectangles 33-35 and rows, 36, and 40 are n-type transistors. The transistors' pins and interconnections are omitted for clarity. Manufacturing DRs dictate that transistors of the same type (e.g., 30-32, rows 37, 38) are restricted to the same row (i.e., same diffusion). Furthermore, p-type and n-type transistors should be placed in groups. The minimum group comprises two transistors. Stacking two tracks or more is possible. In an embodiment of a double stack, shown in Fig.5b, the tracks are inverted to save area. Yet in another embodiment, the tracks are not inverted. Fig. 5b presents a two-track placement design rule where a mirror-like, inversion, placement is implemented to save area.

Fig. 6 presents an embodiment of threshold voltage placement DR constraints. The transistors' pins and interconnections are omitted for clarity. Same-type and same-threshold-voltage transistors are placed side by side (e.g. p-types 41-42, 43-45, and n-types 48-50, 51-53). In an embodiment of the present invention, a minimum group of two same threshold-voltage (Vt) transistors is implemented (e.g. 41-42). A transistor of a single Vt (e.g. 46) must be placed followed by a filler gap (47, 54) or a dummy transistor of the same Vt (right to the HVt transistor 46 for example, HVt designates a high threshold voltage transistor, SVt a standard threshold voltage transistor and LVt a low threshold voltage transistor).

In an embodiment of the present invention, a transistor of any type can be fabricated with one of different threshold voltage (Vt) options (e.g. High Vt, Standard Vt, Low Vt, Ultra Low Vt etc.) and placed according to the relevant design rules constraints.

Fig. 7 presents a fins-based DR placement of the present invention (e.g. for FinFet technology). The transistors' pins and interconnections are omitted for clarity. Fig. 7a and Fig. 7b depict same-type (n-type) same-Vt transistors 55-56 and 59-60 but of different fins. When different fins' transistors are placed side-by-side (i.e. 55-56 or 59-60, abutted) then a filler cell, (57, 58, one grid) or a dummy transistor is placed before any placement of additional transistors from the left side (58) or the right side (57).

Fig. 7c illustrates a best-case DR scenario that results in an optimal placement vs. the case that is depicted in Fig. 7a where same-fin transistors 62-63 are placed to the right.

Fig. 7d presents another placement DR scenario where different-fin transistors 65 and 67 are placed aside a filler cell (66).

In an embodiment of the present invention, each type of transistor may have a variable channel width (including different numbers of fins of FinFET, or stacks of Gate-All-Around or Complementary FET). In an embodiment of the present invention, transistors of different gate lengths including multiple gates (i.e., multiple fingers) are placed and routed.

In an embodiment of the present invention lithography design rules (e.g. “cuts”) are implemented as additional constraints on the placement/or and routing. Yet in another embodiment of the present invention, several layers of metal (e.g. M0, M1, M2…) are implemented for routing.

It is appreciated that certain features of the invention, which are, for clarity, described in the context of separate embodiments, may also be provided in combination in a single embodiment. Conversely, various features of the invention, which are, for brevity, described in the context of a single embodiment, may also be provided separately or in any suitable sub-combination or as suitable in any other described embodiment of the invention. Certain features described in the context of various embodiments are not to be considered essential features of those embodiments unless the embodiment is inoperative without those elements.

Although the invention has been described in conjunction with specific embodiments thereof, it is evident that many alternatives, modifications, and variations will be apparent to those skilled in the art. Accordingly, it is intended to embrace all such alternatives, modifications, and variations that fall within the spirit and broad scope of the appended claims.

All publications, patents, and patent applications mentioned in this specification are herein incorporated in their entirety by reference into the specification, to the same extent as if each individual publication, patent, or patent application was specifically and individually indicated to be incorporated herein by reference. In addition, citation or identification of any reference in this application shall not be construed as an admission that such reference is available as prior art to the present invention. To the extent that section headings are used, they should not be construed as necessarily limiting.

Claims:

1. In an integrated circuit, a method of flattened placement and/or routing of multiple transistors using an Artificial Intelligence (AI) model, comprising the steps of:
   a) receiving an input consisting of a circuit schematic and features;
   b) performing supervised or unsupervised learning of said AI model, to produce a predicted placement and/or routing by said AI model;
   c) using a cost function for optimizing transistor placement and/or routing;
   d) generating an optimal placement and/or routing by:
      d.1) iteratively running cycles of placement and/or routing, based on said AI model;
      d.2) assigning a value to each placement and/or routing combination, according to said cost function and recording the placements and/or routings and value information across the performed iterations until meeting a predetermined metric or exceeding a predetermined number of iterations or elapsed time;
      d.3) ranking each of the generated placements and/or routing according to said cost function; and
      d.4) selecting the k-top (k=1,2,….) placement and/or routing combinations.

2. A method according to claim 1, comprising the steps of:
   a) providing a Neural Network (NN);
   b) providing a Monte Carlo Tree (MCT);
   c) providing a cost function;
   d) receiving an input consisting of a circuit schematic, transistor properties, and a set of design rules;
   e) generating a set of placement and/or routing predictions, for training said Neural Network (NN);
   f) training said Neural Network (NN) with the generated set, to produce a predicted policy and value;
   g) generating an optimal placement and/or routing by iteratively running said Monte Carlo Tree (MCT) in cycles of M playouts being iterations of said MCT, based on the predicted policy and value generated by said neural network (NN);
   h) generating an optimal placement and/or routing by:

d.1) iteratively running cycles of placement and/or routing, based on said AI model;

d.2) assigning a value to each placement and/or routing combination, according to said cost function and recording the placements and value information across the performed iterations until meeting a predetermined metric or exceeding a predetermined number of iterations or elapsed time;

d.3) ranking each of the generated placements and/or routing according to said cost function; and

d.4) selecting the k-top (k=1,2,….) placement and/or routing combinations.

3. A method according to claim 2, wherein any sequence of nodes representing a path down to a leaf in the Monte Carlo Tree (MCT), is a placement combination.

4. A method according to claim 2, wherein only placement combinations of M playouts that meet the design rules are admissible.

5. A method according to claim 2, wherein the cost (reward) function *L* is:

$$L = -\left(\alpha \sum edges + \beta \sum edges\ crossings + \gamma(Total\ Area)\right)$$

where α, β, and ɣ are normalization constants, $\sum edges$ is the total length of wiring, $\sum edges\ crossings$ is the total number of wires' intersections and the *Total Area* is the total area of the placement.

6. A method according to claims 1 and 2, wherein cost functions that comprise a different functional form and different metrics or any combination of area, wiring length, wiring crossings, and other metrics are used.

7. A method according to claim 2, further comprising:
   e) storing the best placement or the k-top placement (k > 1, arbitrary);
   f) carrying out another cycle of M playouts and pruning the nodes that have been already visited in a preceding playout;

g) re-training the NN on the new, pruned tree with the pruned nodes, to produce a policy and a value; and

h) iteratively carrying out the process of placement and/or routing and terminating said process upon reaching a predetermined number of M playouts.

8. A method according to claim 2, wherein the process of optimal placement and/or routing is terminated upon meeting a specific criterion, or after performing a predetermined number of iterations, or after a predetermined run time lapses.

9. A method according to claim 2, wherein an acyclic-directed graph is used instead of a tree structure, to thereby reduce the memory size and improve speed.

10. A method according to claims 1 and 2, wherein the neural network is a single neural network or comprises a combination of multiple neural networks of the same or different types.

11. A method according to claims 1 and 2, wherein a Neural Network (NN) is selected from the group of:
- Graph Neural Networks for inputs embedding;
- Convolutional Neural Network for placement embedding;
- Dense neural network for feature embedding;
- Dense neural network for policy and value generation.

12. A method according to claims 1 and 2, wherein a netlist or an RTL code serves as an input to synthesize a netlist.

13. A method according to claims 1 and 2, wherein the cost function reflects the total area, the total wiring, or both, said cost function being any function that minimizes the placement area and or the wiring length.

14. A method according to claims 1 and 2, wherein the cost function is any functional form of one or more of the following:

- the circuit area;
- wiring length;
- wires crossing;
- specific design rules;
- transistor properties.

15. A method according to claim 2, comprising:
    a) inputting the circuit netlist and its transistors' properties to a Graph Neural Network (GNN) that generates a two-dimensional (2D) features array of size N x h, where N is the total number of transistors and h is a hidden layer size;
    b) flattening the 2D array of size N x h by reducing its dimension, and feeding the flattened 2D array to MLP1, which outputs a vector of size m;
    c) feeding the convolutional neural network with a vector of Unique Identification Numbers (UIN) of each transistor's position in the preceding placement and/or routing prediction and generating an output vector of size n;
    d) combining the vectors of size m and n and feeding the concatenated vector into a dense NN, to generate a new policy and value.

16. A method according to claim 2, wherein the outputs of neural networks MLP1 and CNN, are concatenated.

17. A method according to claim 2, wherein the output of the neural network is a single output.

18. A method according to claim 2, wherein the neural network generates a policy only; inferring a policy that maximizes the reward, or a value only to infer the value of a predicted placement.

19. A method according to claim 2, wherein the transistors' properties are a unique identification number (UIN) of the position, transistor type, a threshold voltage, the width or the number of fins, a gate length or the number of fingers.

20. A method according to claim 2, wherein a partial list of the transistors' properties or an extended list with additional properties is used as input to the neural network.

21. A method according to claim 2, wherein the input to the neural network is a concatenated vector of the transistor UINs.

22. A method according to claims 1 and 2, wherein the input to the neural network is a two-dimensional array of the transistors' X and Y positions.

23. A method according to claims 1 and 2, wherein the input to the neural network is any type of data format that contains the transistors' position information such as an image file.

24. A method according to claims 1 and 2, wherein the initial weights of the neural networks are pre-trained on a smaller circuit of fewer transistors, uniformly distributed, or determined according to methods that pertain to AI.

25. A method according to claim 2, wherein a UIN is a decimal or a binary number representation of a transistor position in the placement, or a one-hot encoding representation of a transistor position in the placement, wherein any unique representation may serve as a UIN.

26. A method according to claims 1 and 2, wherein each type of transistor has a variable channel width, including different numbers of fins of FinFET, or stacks of Gate-All-Around or Complementary FET.

27. A method according to claim 2, wherein the predetermined score is the standard Upper Confidence Bound (UCB) score.