Deep Reinforcement Learning-Based Power Distribution Network Design Optimization for Multi-Chiplet System

Weiyang Miao*
School of Electrical and Electronic Engineering
Nanyang Technological University
Singapore
MIAO0040@e.ntu.edu.sg
(**Attached student at Institute of Microelectronics*)

Zhen Xie*
School of Electrical and Electronic Engineering
Nanyang Technological University
Singapore
ZHEN004@e.ntu.edu.sg
(**Attached student at Institute of Microelectronics*)

Chuan Seng Tan*
School of Electrical and Electronic Engineering
Nanyang Technological University
Singapore
tancs@ntu.edu.sg
(**Joint appointment at Institute of Microelectronics*)

Mihai D. Rotaru
Institute of Microelectronics
*Agency for Science, Technology and Research (A*STAR)*
Singapore
mihai_dragos_rotaru@ime.a-star.edu.sg

***Abstract*—This study employs a reinforcement learning (RL) technique, Dueling Double Deep Q Network (DDDQN), to optimize the power distribution network (PDN) in a high-density fan-out wafer-level package (HD-FOWLP) for a four-chiplet system, addressing the complexities of the PDN design with multiple power domains. A four-chiplet system is arranged in a ring topology, with each chiplet featuring seven distinct power islands, resulting in a total of 28 separate power islands within the power layer of the HD-FOWLP package's redistribution layer (RDL). The reinforcement learning algorithm used in this work addresses the difficulties of expansive design space and offers efficient optimization as well as shorter development time. A grid of 13 × 13 cells (in the format of a matrix) was used as the design environment to represent one quadrant of the package PDN; the design rules were translated into analytical expressions as the reward functions for RL. Training the RL model to generate a viable solution took 6 CPU hours. The proposed model demonstrates the ability to reduce development time and cost, easing engineers' workload and development difficulty.**

Keywords—chiplet, heterogenous integration, power distribution network, power integrity, reinforcement learning

I. Introduction

With the rapid development of science and technology, applications such as 5G/6G, artificial intelligence, cloud computing, autonomous driving, and big data have boosted the development of stronger and faster chips to meet the demand for higher performance, energy efficiency, and reliability. However, Moore's Law can no longer be applied to the development of system-on-chip (SoC) due to expensive research and development cost, long design cycles, and technological bottlenecks. In order to continue Moore's Law, many novel approaches have emerged, and some of the strong contenders include heterogeneous integration, 2.5D/3D advanced packaging, and chiplets. The concept of multi-chip modules (MCM) has been around for many years, however it became a growing trend only after the maturity of advanced packaging and interconnect interface. Although chiplet brings us larger systems and lower development costs, the challenges arise accordingly, such as high-speed interconnect design, power delivery, thermal management, larger design space, etc. Especially for the design of power integrity, a single chiplet already has a number of domains with different voltage levels. If several chiplets are packaged together, the number of different voltage islands will again increase several times, which brings a great challenge to the design of power distribution network (PDN) for multiple chiplets at the package level.

The pressing need to address this escalating complexity of chip design drives the integration of machine learning and semiconductor design, the adoption of machine learning methodologies is motivated by many advantages they offer, these include rapid exploration of complex design options, efficient optimization, and the ability to uncover the intricate patterns within extensive design data [1]. However, despite these merits, an inherent challenge persists in the form of limited access to open-source training data.

Reinforcement learning (RL), a subfield of machine learning, is a solution to surpass this data limitation. Reinforcement learning empowers algorithms to explore the design space autonomously, find optimal design solutions, and iteratively update their strategies. Reinforcement learning introduces a paradigm shift in chip design by eliminating the dependence on extensive training dataset [2]. Some works used neural-network-based supervised learning models to design the power distribution network, however, it is only for the single SoC PDN design instead of multi-chiplet systems, and they had to generate their own training data under certain design constraints, so the AI model trained can only be task-specific, which cannot achieve cross-design [3]. Other works used reinforcement learning to design the PDN for silicon interposer based 2.5D/3D packaging, but they only focused on the decoupling capacitor design optimization, which is just one part of the overall power distribution system and didn't consider the power route of voltage supply (VDD) and ground (GND) [4, 5].

In this paper, a deep reinforcement learning-based framework is proposed to optimize the package-level power

distribution network design (including power plane and power route) for a multi-chiplet system assembled on an advanced 2.5D packaging called high density fan-out wafer level package (HD-FOWLP) [6]. The proposed method outperformed in terms of computing time, it reduces the full search simulation that takes more than a month by an engineer to about six hours and optimizes the solution from a large design space up to $10^{17} – 10^{18}$ states. This approach solves the time-consumption problem of design optimization by humans and overcomes the lack of open-source training data in the field. Next, the details of design environment and reinforcement learning algorithm will be described, and the results verification of the proposed model against the solution designed by humans will also be shown.

II. Methodology

A. Problem Statement

In this work, a deep reinforcement learning approach designed for optimizing the power distribution network in multi-chiplet package is introduced. The objective is to assign the different power domains to each unit cell of the package PDN one by one and tile the whole power plane, to get a reasonable distribution of power islands as well as a low PDN impedance response to improve the power integrity design.

For a design task like this, it is not easy to find open source training data or generate your own training data, which is why reinforcement learning is used in this project. Reinforcement learning does not require training data because it can explore the design space by itself; it gets reward for good actions and gets penalty for bad actions, just like playing a game. A reinforcement learning technique called Dueling Double Deep Q Network (DDDQN) [8] was implemented, the RL agent is in the form of a deep neural network as shown in Fig. 1, it will interact with the design environment and learn to perform the best actions to maximize the reward. One important concept is that the agent always chooses the action with the highest Q-value among all possible actions, this Q-value not only estimates the immediate reward of current action, it also predicts the sum of rewards for all the future actions. There are three primary research questions:

1) How to formulate the package PDN design environment as a reinforcement learning problem?

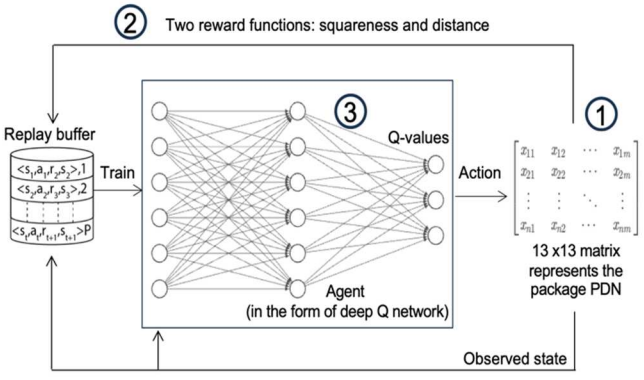


Fig. 1 Schematic of proposed reinforcement learning model.

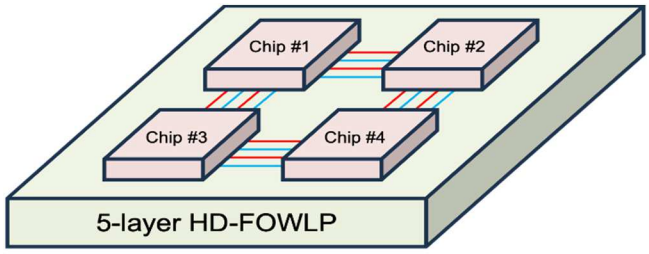


Fig. 2 Multi-chiplet system assembled on high density fan-out wafer level package.

2) What kind of rewards should be given to the agent?

3) Which kind of reinforcement learning algorithm should be used?

The paper is organized as follows. Section II-B provides the overview of design problem and the formulation of the design problem as a reinforcement learning problem. Sections II-C and II-D describe the details of action space and reward function representations. Section II-E describes the architecture of dueling double DQN.

B. Design Problem and State Space

The system used as a test case in this work has been extensively discussed previously in [6] and [7] and consists of 4 accelerator chiplets interconnected through AIB in a ring topology integrated in a FOWLP package. Fig. 2 shows the architecture of a 4-chiplet AI accelerator that used as testbed in this work; the four chiplets are connected in a ring topology. The Intel Advanced Interface Bus (AIB) is used to transmit the signal from one chiplet to another. Fig. 3 shows the footprint details of one chiplet, the C4 bumps and micro-bumps are distributed across the chiplet to supply power and ground, connect the AIB I/Os between chiplets, and route out 27 I/Os to next level of package. There are seven voltage domains in one chiplet: VDDC1, VDDC2, VDDC3, VDDK1, VDDK2, VDDTR, and VDDIO. The allocation of C4/micro bumps for chiplet is fixed

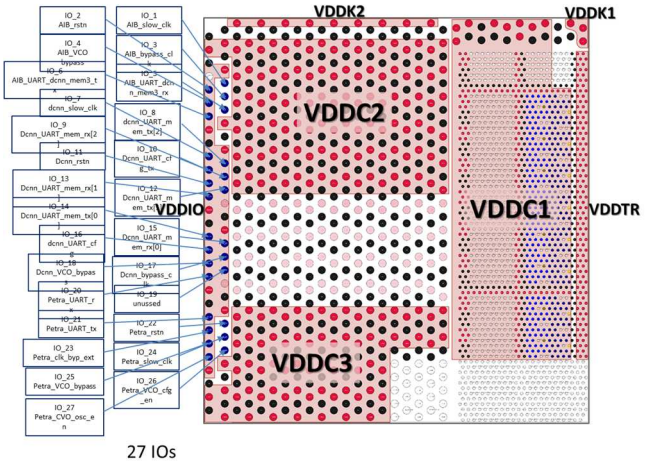


Fig. 3 Chiplet footprint with seven different power domains [6].

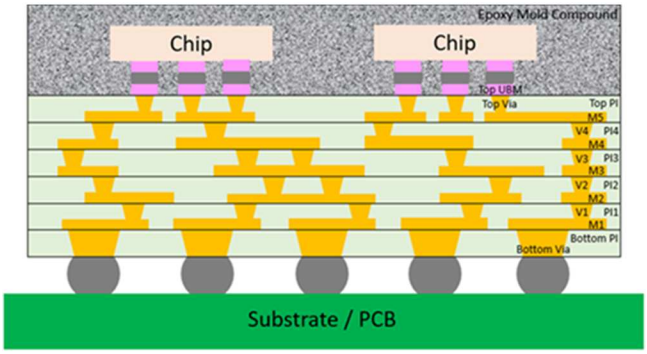


Fig. 4 HD-FOWLP 5 layer stack-up [6]

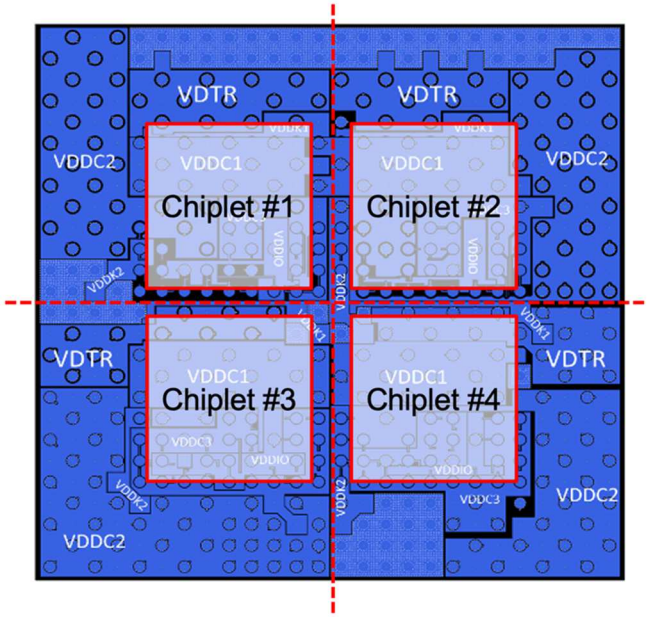


Fig. 5 Distribution of power domains in package PDN [6].

because the design of the chiplet has already been finalized. The proposed RL model uses this allocation as input information to generate the package PDN design.

The cross section view of package for the 4-chiplet system is shown in Fig. 4. It consist of a stack-up with 5 redistribution layers (RDL). The top layer M5 consists of under bump metal (UBM) pads that connect to the C4/micro bumps of the chiplet; M3 and M4 are used to route the AIB interconnect between chiplets; M1 and M2 are used for the package level power distribution network; M2 is the ground plane and M1 is the power plane. There are 7 power domains for one chiplet, therefore 28 power domains need to be partitioned on the M1 layer for a 4-chiplet system as shown in Fig. 5.

Below M1 are 26 × 26 solder balls that route out the power/ground and signal I/O to the next level of the package; the pitch of solder ball is 500μm and the total size of the package is 13.5mm × 13.5mm. The package power plane (M1) can be divided into 26 × 26 unit cells according to the pitch of the solder

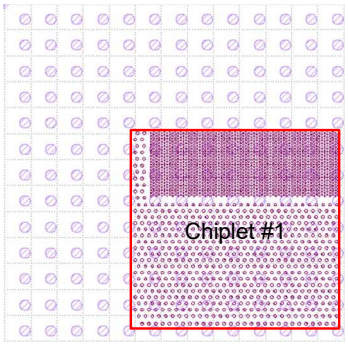


Fig. 6 One chiplet and its corresponding quadrant of power plane in package-level PDN.

ball. The RL model was trained to generate the package power plane for one quadrant (one chiplet) at a time, to save training time and design space for the algorithm. Therefore, a 13 × 13 matrix can be used to represent the one quadrant of PDN with 13 × 13 unit cells for one chiplet as shown in Fig. 6. Each cell can be assigned to one of the seven power domains.

C. Action Space

The RL agent assigns power domains for one unit cell at a time, the actions are all possible power domains that the current unit cell can be assigned without violating the maximum usage of the power domain and other design constraints. For one quadrant of PDN with 13 × 13 unit cells, the RL agent needs to take 169 actions to finish the PDN design. There are several ways to define the action set, for example action set can be all combinations of power domain selections and coordinates for all unit cells, which gives us an action space with 7×13×13 = 1183 possible choices of actions. Each time the RL model uses the current state as input and chooses an action based on the current policy, the agent will take the action and the state will move to the next state as shown in Fig. 7.

However, such a way of defining action space can make the training process slow and inefficient. In our approach, the sequence in which the unit cells are assigned by the agent is predefined, the agent will move through the unit cells line by line and select the power domain for the current cell. Therefore, the coordinate variable of unit cells will be removed from the

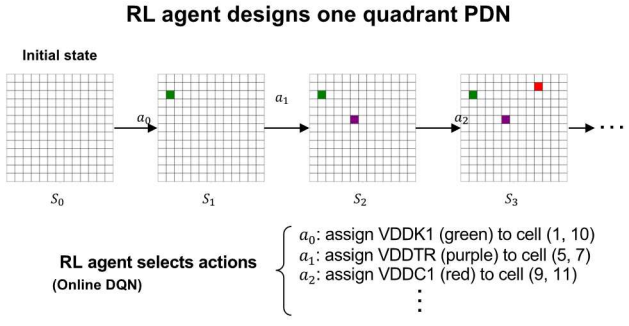


Fig. 7 PDN design process performed by reinforcement learning model.

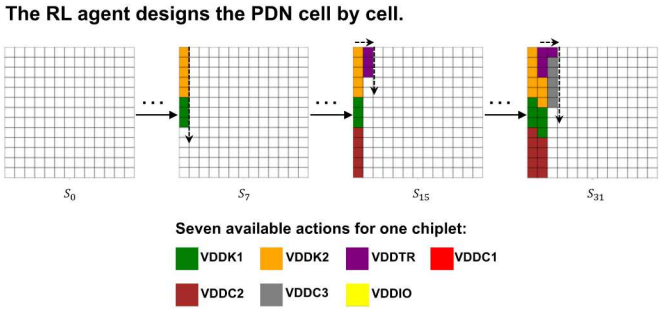


Fig. 8 Proposed PDN design process.

action space, leaving the action set with only 7 possible actions, which are choices of seven power domains (Fig. 8). Such simplification of the action set can reduce RL model training time from days to hours.

D. Reward Function

During the training process of the RL model, rewards are given for all actions except the last action in an episode, where the rewards are used to measure the value of actions performed. The design of the reward function is a critical aspect of RL as it translates the design rules into analytical expressions for a given design task. Two main reward functions are used in our RL model: the squareness of a power island, and the distance between the power island to its corresponding chiplet power bumps. The experimental results of the RL model demonstrate good reliability of these two reward functions. Usually, the EDA simulation for design evaluation can only be conducted after the whole PDN is completely designed and fully connected, but the reward function allows the agent to get a preliminary evaluation as soon as a unit cell (i) has been designed, even if the overall PDN is incomplete and unconnected.

A squareness reward is defined in our approach, it measures the geometric shape of a power island in package PDN. Squareness refers to the quality of being square, if a power island consists of 16 cells, then a 4×4 arrangement will have a higher squareness reward than a 2×8 arrangement. The considerations and design of this squareness reward are derived from the results of previous work [6] as shown in Fig. 9, designs of the VDDC2

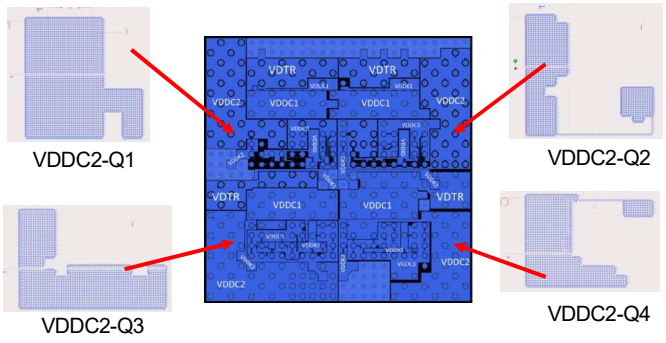


Fig. 9 PDN modeling for the VDDC2 in the four quadrants of the package.

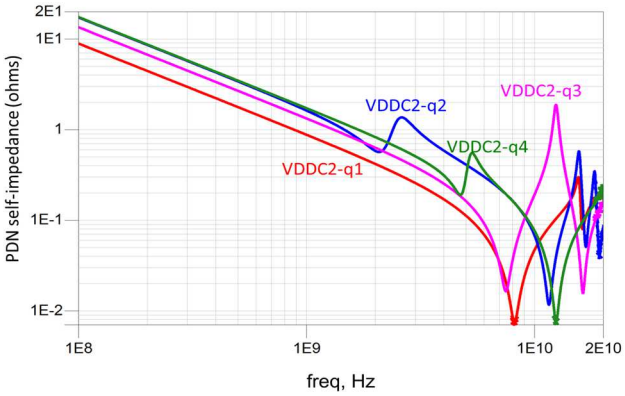


Fig. 10 Simulated PDN self-impedance for the VDDC2 from four quadrants [6].

package power island are different for each quadrant, and the impedance response is shown in Fig. 10. The VDDC2 in quadrant 1 with a wider structure has the lowest self-impedance due to the lower inductance of the power island, the other three VDDC2 have similar areas, however due to their layout (narrower structures), they have larger inductance. In other words, VDDC2 in quadrant 1 has a higher squareness reward because it is shaped more like a square, and a square geometry is more effective in avoiding narrow connections thereby reducing inductance and impedance. In our approach, when a cell is assigned to a power island, the agent checks the four cells adjacent to the current cell, and if the adjacent cells belong to the same power island, then the squareness reward is given according to the number of the same cells (Fig. 11). In order to standardize this reward, the reward is divided by 4, so that the reward is ranged from 0 to 1. This method of defining reward forces the RL model to cluster the cells with the same power island together to get more rewards and therefore design power islands with wider structure and lower impedance.

Distance is another reward that is equally important as squareness reward. Fig. 12 demonstrates one quadrant of package power plane (undesigned) and C4/micro power bumps allocations that are connected to one chiplet. The package power

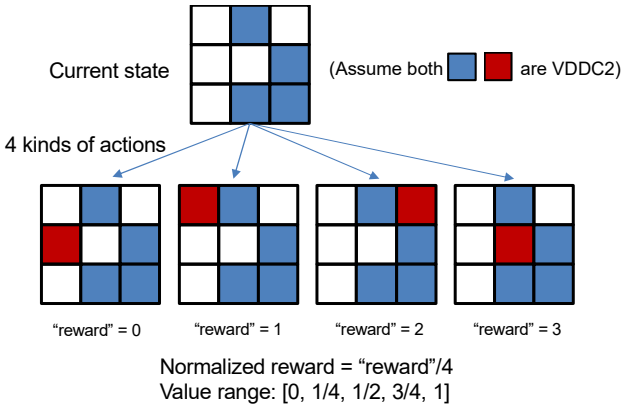


Fig. 11 Calculation of squareness reward.

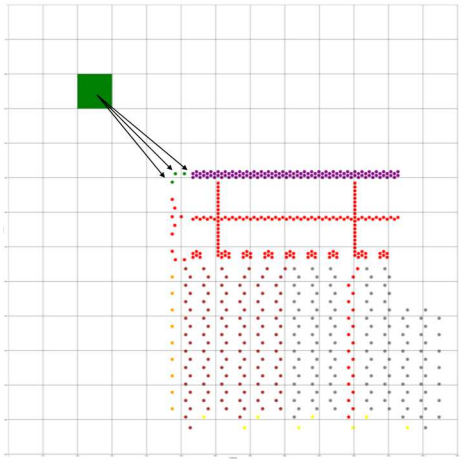


Fig. 12 Mechanism of distance reward.

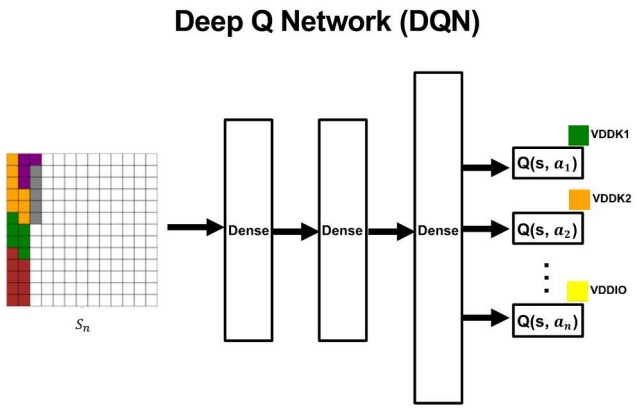


Fig. 13 Structure of Deep Q Network.

plane is represented by a grid with 13 × 13 cells, and bumps with different colors indicate they are connected to seven different power domains of chiplet. For example, the three green bumps are for the power domain of VDDK1, and the RL agent is assigning the current cell to the VDDK1 power island (painted green). A cell can deliver power more efficiently when it is closer to the corresponding bumps because a closer distance results in less resistance and less inductance, so in our distance reward function, a shorter distance will give the agent a higher reward. In our approach, the distance reward is calculated by getting the Manhattan distance from the current cell to each of the three bumps, normalizing these distance values to make them range from 0 to 1, and subtracting these values from 1 to ensure that shorter distance will have a higher reward, and finally calculating the average value for three bumps, which is the distance reward for the current cell placement.

E. Structure of Reinforcement Learning Algorithm

Deep Q-Network (DQN) is an advanced reinforcement learning algorithm developed by DeepMind researchers [9], it applies a deep neural network to approximate the Q-function, where the input is the current design state, and the output is the Q-values estimation for each possible action at current state as shown in Fig. 13.

Double DQN (DDQN) is an extension of DQN that addresses one of the limitations of the original DQN algorithm [10]. In DQN, during the target Q-value estimation, the same neural network is used for both selection and evaluation of the action. This can lead to the overestimation of Q-values, which may result in a suboptimal policy. DDQN introduces the idea of decoupling action selection and evaluation. It maintains two separate networks, one for action selection (online network) and another for Q-value estimation (target network). This decoupling helps reduce overestimation bias and leads to more stable and improved performance.

Dueling Double DQN (DDDQN) combines the benefits of both Dueling DQN and Double DQN concepts. Dueling DQN is the modified version of DQN, it separates the Q-value prediction into two streams: one predicts the value of current states and another one predicts the advantage of taking different

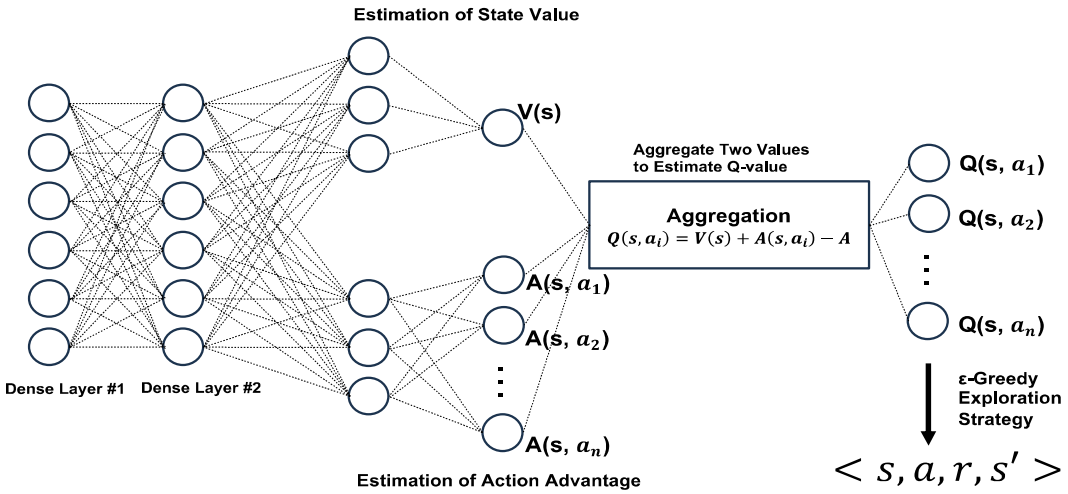


Fig. 14 Online DQN for training data generation.

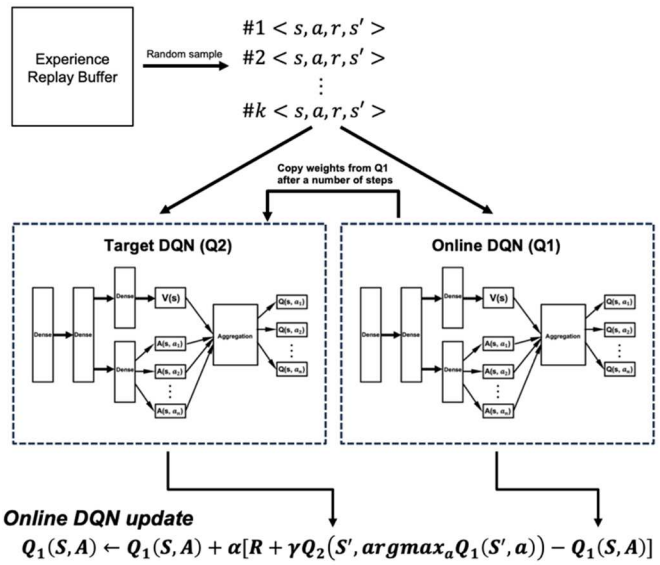


Fig. 15 RL model learning from the training data.

actions. Such structure can improve learning efficiency. Dueling DDQN then incorporates this architecture into the Double DQN framework. By doing so, it addresses overestimation bias while also providing a more efficient way to estimate the value and advantage functions.

There are two steps to training the RL model, which are training data generation and policy update. For the training data generation (Fig. 14), the RL agent (online DQN) interacts with the design environment, it uses the current state (s) as input for the online network and estimates the Q-values for each action. Then the agent uses an epsilon-greedy strategy to select the action (a) and perform this action to move to the next state (s'). A set of data < s, a, r, s'> is generated from this step and stored in the experience replay buffer [11].

Training and learning are synchronized, a mini-batch of data is randomly sampled from the experience replay buffer, the online DQN and target DQN are used together to process this batch of data to update the RL model policy (weight of the online DQN network neurons) as shown in Fig. 15. In summary, one loop is that the RL agent (online DQN) interacts with the design environment and collects the training data, and the other loop processes the training data with the online DQN and target DQN to compute the loss function and updates the weights of the Online DQN.

III. Results and Discussion

A. Hyperparameters of RL Model

After several attempts and experiments, a suitable set of hyperparameters was found for the PDN design as shown in Table I. The input size of the model is the same as the size of design environment, which is 13×13; two dense layers have 256 neurons and 128 neurons respectively. The size of output layer as well as the intermediate advantage layer are set to number of the available actions which is 7. Replay buffer size and mini-batch size are set to 3000 and 200, discount factor is set to 0.5 and learning rate of optimizer is set to 0.001. Value of epsilon

TABLE I. RL Model Hyperparameters

	Hyperparameter	Value
RL DQN Structure	Input layer	13 × 13
	Dense layer 1	256
	Dense layer 2	128
	Output layer	7
Training process	Replay buffer size	3000
	Mini-batch size	200
	Discount factor	0.5
	Learning rate	0.001
	Target update frequency	500
	Epsilon	1 → 0.01

was gradually reduced from 1 to 0.01 and 0.01 was used throughout the episodes until the end of training. The frequency of target DQN update is set to 500, for every 500 times that the online network has trained and learned, the weights of target DQN are replaced with weights of online DQN.

B. RL Training Results

There are 1500 training episodes for one quadrant of PDN design, the learning curve for quadrant 1 is shown in Fig. 16, which demonstrates good convergence and high episode return of our proposed RL model structure and hyperparameters. The episode return increases along the iterations and gradually converges to an optimal level.

The original PDN design for the chiplet system is shown in Fig. 17 (a), it took about three months to design and verify by engineers. The AI-generated PDN design is shown in Fig. 17 (b), it took 6 CPU-hours to train the RL model and generate the viable PDN design for one quadrant. Normally, 1 GPU hour is equivalent to 4 CPU hours, if a proper GPU is used for the project, the training speed can be further increased by 4 times. The AI-generated solution also makes good use of all areas of

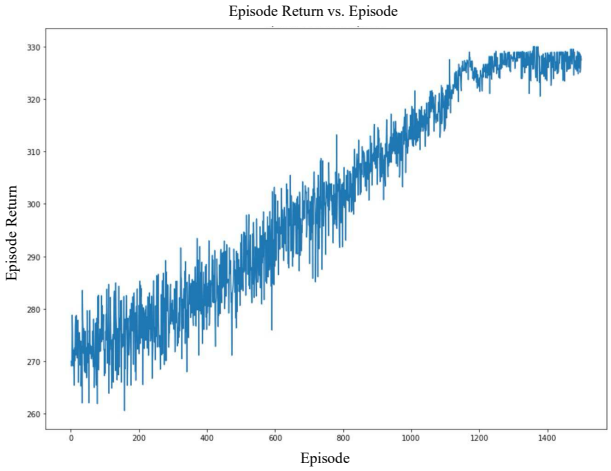


Fig. 16 Learning curve of episode return.

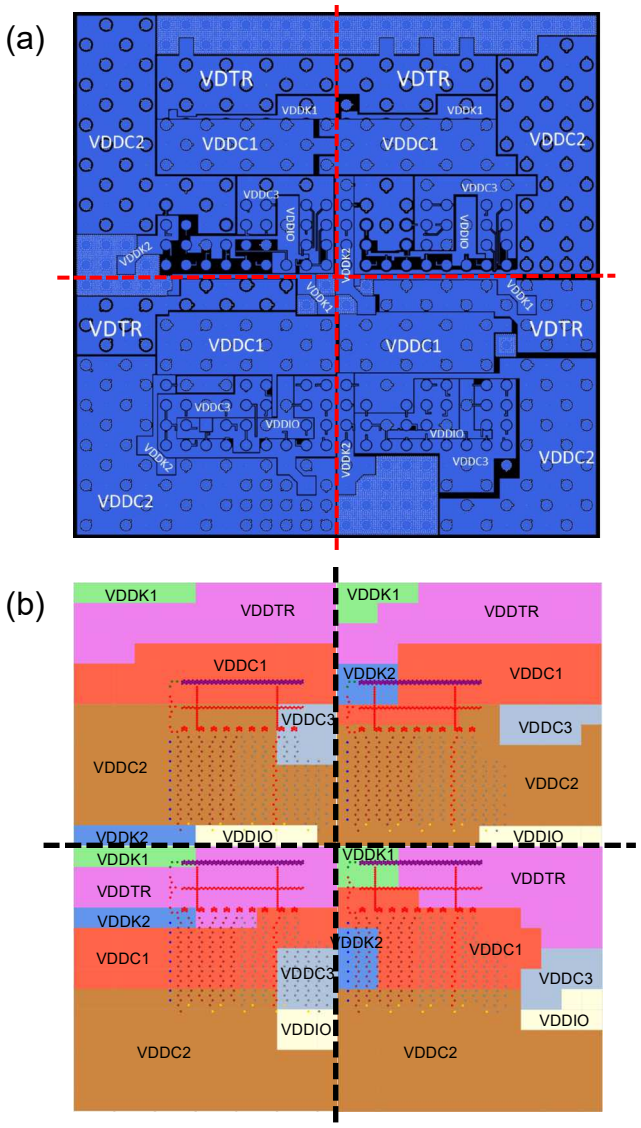
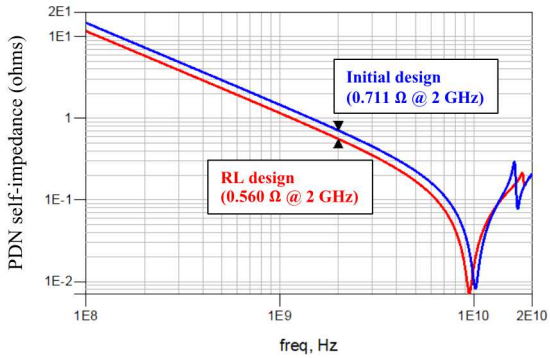


Fig. 17 PDN designs comparison: (a) engineer generated solution, (b) RL model generated solution.

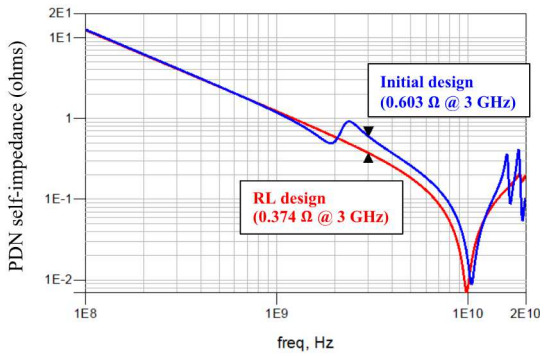
package power plane and leaves no residual cell. By implementing this RL model, it can reduce the development time and cost, as well as ease engineers' workload and design difficulty.

C. *Impedance Response*

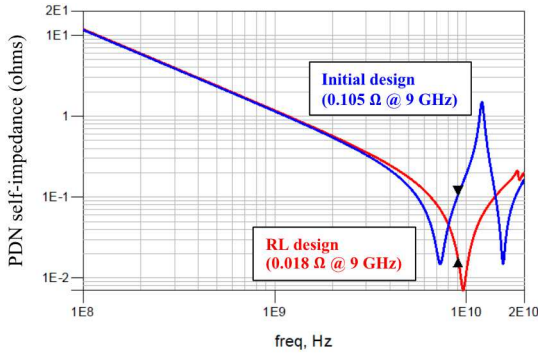
PDN modeling and simulation are conducted to verify the design generated by RL. Take VDDC2 for example, Fig. 18 shows the impedance comparison between engineer's design and RL's design for power domain of VDDC2 in four quadrants. In quadrant 1, the impedance is reduced by 21.2% at 2 GHz; in quadrant 2, the impedance is reduced by 38.0% at 3 GHz; in quadrant 3, the impedance is reduced by 82.9% at 9 GHz; in quadrant 4, the impedance is reduced by 66.7% at 5 GHz. The impedance reduction at most frequency points proves that the proposed reinforcement learning method is feasible, the RL model settings and choice of reward functions can generate a



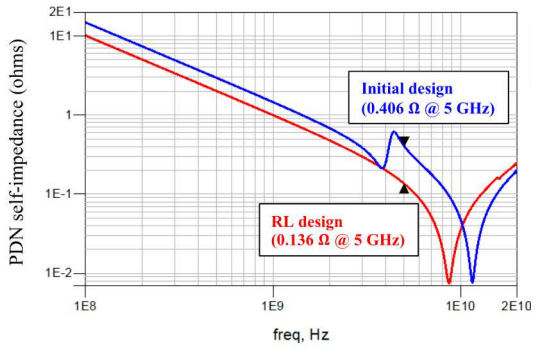
(a) Impedance comparison for VDDC2 in quadrant 1



(b) Impedance comparison for VDDC2 in quadrant 2



(c) Impedance comparison for VDDC2 in quadrant 3



(d) Impedance comparison for VDDC2 in quadrant 4

Fig. 18 Self-impedance comparisons between RL and original design for VDDC2 in four quadrants.

PDN design that is comparable to engineer-generated solution, but with a much shorter design time.

IV. Conclusion

In this paper, a deep reinforcement learning-based framework is introduced to design the package-level power distribution network for a multi-chiplet system with 28 power domains. The proposed AI model can generate a viable PDN design with a much shorter design time. The new proposed PDN structure also has a lower self-impedance as compared to the original design.

For any kind of reinforcement learning application, defining the design environment, reward function, and structure of the RL model are the most critical steps. In this work, squareness and distance are used as reward functions; the design and verification results of PDN also prove that the choice of this reward mechanism is reasonable. The proposed method can be used for future PDN design of new structures or platforms, if there are new design rules and design constraints, new reward functions can be implemented to upgrade the reinforcement learning model.

V. Acknowledgment

This research was supported by the Agency for Science, Technology and Research (A*STAR) under Applied Centre of Excellence in Advanced Packaging 3.0 (Grant No. I2101E0008).

References

[1] M. Swaminathan, H. M. Torun, H. Yu, J. A. Hejase and W. D. Becker, "Demystifying Machine Learning for Signal and Power Integrity Problems in Packaging," in IEEE Transactions on Components, Packaging and Manufacturing Technology, vol. 10, no. 8, pp. 1276-1295, Aug. 2020.

[2] A. F. Budak, Z. Jiang, K. Zhu, A. Mirhoseini, A. Goldie and D. Z. Pan, "Reinforcement Learning for Electronic Design Automation: Case Studies and Perspectives: (Invited Paper)," 2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC), Taipei, Taiwan, 2022, pp. 500-505.

[3] V. A. Chhabria and S. S. Sapatnekar, "OpeNPDN: A Neural-Network-Based Framework for Power Delivery Network Synthesis," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41, no. 10, pp. 3515-3528, Oct. 2022.

[4] H. Park et al., "Deep Reinforcement Learning-Based Optimal Decoupling Capacitor Design Method for Silicon Interposer-Based 2.5-D/3-D ICs," in IEEE Transactions on Components, Packaging and Manufacturing Technology, vol. 10, no. 3, pp. 467-478, March 2020.

[5] H. Park et al., "Transformer Network-Based Reinforcement Learning Method for Power Distribution Network (PDN) Optimization of High Bandwidth Memory (HBM)," in IEEE Transactions on Microwave Theory and Techniques, vol. 70, no. 11, pp. 4772-4786, Nov. 2022.

[6] M. D. Rotaru, W. Tang, D. Rahul and Z. Zhang, "Design and Development of High Density Fan-Out Wafer Level Package (HD-FOWLP) for Deep Neural Network (DNN) Chiplet Accelerators using Advanced Interface Bus (AIB)," 2021 IEEE 71st Electronic Components and Technology Conference (ECTC), San Diego, CA, USA, 2021, pp. 1258-1263.

[7] T. Chou; W. Tang; M. D. Rotaru; C. Liu; R. Dutta; S. Lim; D. Ho; S Bhattacharya and Z. Zhang, "NetFlex: A 22nm Multi-Chiplet Perception Accelerator in High-Density Fan-Out Wafer-Level Packaging," 2022 IEEE Symposium on VLSI Technology and Circuits, Honolulu, HI, USA.

[8] Z. Wang, et al., "Dueling network architectures for deep reinforcement learning," International conference on machine learning, PMLR, 2016.

[9] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," Nature 518, 529–533 (2015).

[10] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in Proceedings of the AAAI conference on artificial intelligence, vol. 30, 2016.

[11] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," arXiv preprint arXiv:1511.05952 (2015).