Template-based PDN Synthesis in Floorplan and Placement Using Classifier and CNN Techniques

Vidya A. Chhabria[1], Andrew B. Kahng[2], Minsoo Kim[2], Uday Mallappa[2], Sachin S. Sapatnekar[1], and Bangqi Xu[2]
[1]University of Minnesota; [2]University of California, San Diego

***Abstract*—Designing an optimal power delivery network (PDN) is a time-intensive task that involves many iterations. This paper proposes a methodology that employs a library of predesigned, stitchable templates, and uses machine learning (ML) to rapidly build a PDN with region-wise uniform pitches based on these templates. Our methodology is applicable at both the floorplan and placement stages of physical implementation. (i) At the floorplan stage, we synthesize an optimized PDN based on early estimates of current and congestion, using a simple multi-layer perceptron classifier. (ii) At the placement stage, we incrementally optimize an existing PDN based on more detailed congestion and current distributions, using a convolution neural network. At each stage, the neural network builds a safe-by-construction PDN that meets IR drop and electromigration (EM) specifications. On average, the optimization of the PDN brings an extra 3% of routing resources, which corresponds to a thousands of routing tracks in congestion-critical regions, when compared to a globally uniform PDN, while staying within the IR drop and EM limits.**

I. Introduction

Power delivery network (PDN) design is highly constrained due to limited voltage headroom, high on-chip simultaneous switching noise, high wire resistances, and high switching currents. PDNs compete for scarce on-chip wiring resources with signal and power nets [1], and PDN planning throughout the design cycle is key to managing interconnect resources, aiding design closure. Prior work on PDN optimization has been constrained by the large size of the power grid, which may contain millions to billions of nodes. Optimization must invoke circuit analysis, which involves the solution of a large system of equations, $G\mathbf{V} = \mathbf{J}$, where G is the conductance matrix for the PDN, $\mathbf{J}$ is the vector of current sources, and $\mathbf{V}$ is the set of voltages at each node in the PDN. Several techniques for solving PDNs have been explored, based on multigrid methods [2], exploiting hierarchy [3], [4], or working in the frequency domain [5].

An optimized PDN must satisfy several specifications: IR drop constraints bound the allowable voltage drop from the pads to each node; electromigration (EM) constraints limit the maximum current density in wires; and congestion constraints balance PDN routing resources with contending signal/clock interconnects. Thus, optimization must invoke PDN analysis in its inner loop, which makes it highly computational [1], [6]–[8]. To avoid this, [9] proposes a methodology for PDN synthesis at the placement stage, and constructs a uniform grid across the chip that meets an IR drop specification. In this case, the uniform grid is possibly sub-optimal (over-designed) when compared to an irregular PDN constructed based on the variations in current and congestion distributions across the chip. As against the extremes of the conventional *ad hoc* irregular PDN or this fully uniform PDN, we propose a structured yet flexible approach to PDN design that places several restrictions on the optimization search space:

(i) We use unidirectional wires in every metal layer. This is consistent with design rules for FinFET nodes, where layout restrictions dictate gridded layout with strict directionality requirements. The power grid in the lower layers (M1/M2) lines up with the standard cells and is already regular. We maintain this regularity over all utilized layers.

(ii) Rather than allowing arbitrary combinations of pitches over all layers, we limit the choices to a few fixed *templates*. The metal pitch for each template is constant in each metal layer, but may vary across layers. Pitches are chosen so that the templates can be readily abutted.

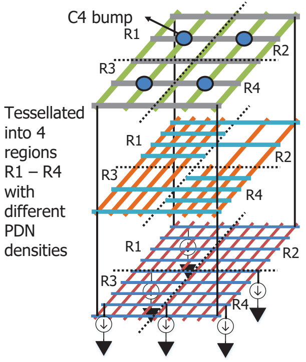
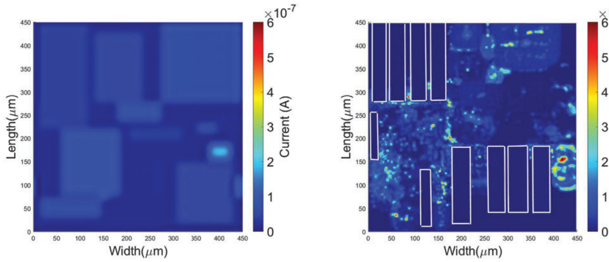

C4 bump
R1
R2
R3
R4
Tessellated into 4 regions R1 – R4 with different PDN densities
R1
R2
R3
R4
R1
R2
R3
R4

Figure 1: A template-based PDN with piecewise uniform pitches.

Specifically, this work tessellates the entire chip into rectangular *regions* (Fig. 1), each of which uses one of the templates; different regions may use different templates. This resembles the idea of locally regular, globally irregular grids in [10] for two top-level metal layers, but we (i) use a limited set of templates, and (ii) apply this approach across the entire interconnect stack. The PDN design problem then reduces to mapping templates to regions of the chip.

A structured template-based grid in itself does not overcome the computational bottleneck of PDN analysis: the grid still has a very large number of nodes. Our PDN optimizer completely bypasses an expensive PDN analysis step in the inner loop by building an ML model to construct the optimal template set, for a given distribution of currents drawn from the grid and a distribution of congestion. *The major insight of this work is that by pushing analysis into a one-time training step, we can train a neural network to synthesize a PDN with very low computational cost in the inference step.*



(a)

(b)

Figure 2: Current distribution of Core1 at the (a) floorplan block-level granularity and (b) detailed granularity after global placement. The design has 171,697 cells and 10 macros and is designed in a foundry 16nm FinFET (16FF) technology. Although the latter has more hot spots, the early floorplan-stage prediction provides a good estimate of the average current distribution.

Our approach consists of two types of neural networks, one applicable to the early floorplanning stage of design and another

This work is supported in part by the DARPA IDEA program as a part of the OpenROAD project

for the later placement stage. These stages differ in the amount of design detail that is available for PDN design. The neural networks are devised to operate *self-consistently* so that placement-stage PDN design corresponds to an incremental refinement, i.e., a small perturbation of floorplan-stage design. This provides predictability in PDN congestion, which aids design closure. Both neural networks are trained to synthesize a *safe-by-construction* PDN.

Fig. 2 shows the current distributions at the floorplan and placement stages for a RISC-V core [11] (Core1). From the figure, it is seen that the underlying data is of coarser granularity at the floorplan stage than at the placement stage. Here, the PDN can be synthesized using a simple multilayer perceptron (MLP) classifier. At the placement stage, current distributions involve higher-dimensional fine-grained data; therefore, PDN synthesis at this stage requires a more complex convolution neural network (CNN) that navigates this complexity.

An overview of our ML-based flow, which uses a set of user-defined PDN templates (Section II), is illustrated in Fig. 3. The training flow (left) is a *one-time* expense that is performed offline; given a large training set, it uses a high-quality simulated annealing (SA) heuristic to find an optimized template for a given current and congestion distribution (Section III). In the inference flow (right), for any design the slow SA-based optimizer is replaced by a trained CNN or MLP inference engine (depending on the stage in the design flow) that can rapidly synthesize the optimal template for each region of a design. We evaluate our proposed methodology on example circuits in Section IV.

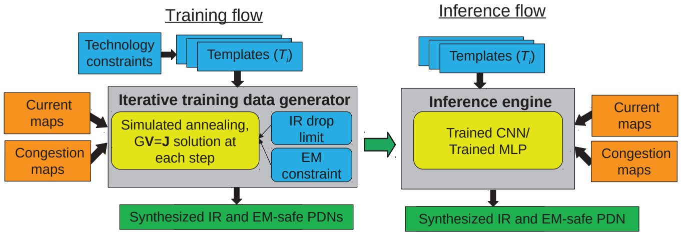


Figure 3: PDN optimization scheme: The training flow produces the "golden" data and trains the neural network, while the inference flow uses the trained network to synthesize the PDN.

II. PDN TEMPLATES

A. Template Definition

Each template uses a constant wiring pitch in each layer, though the pitch may vary across layers. The selection of templates must be cognizant of the factors that influence PDN wiring resources:

- The design rules on each metal layer dictate the pitch (stripe width, stripe spacing between consecutive stripes), metal density, via densities, and the preferred direction (horizontal/vertical).
- The spatial distribution of currents drawn from the PDN influences the required wire density in the PDN.
- The signal/clock routing congestion in each region of the chip constrains the resources available to the PDN.

VDD/VSS package bumps are distributed uniformly over the chip.

A critical requirement in the construction of the PDN templates is their *stitchability*, i.e., if two templates are placed side by side, they should align at the edges. In each layer, if the pitches of the PDN stripes are an integer multiple of the minimum track spacing in that layer, the wires are well connected to each other at the edges of each template, though perhaps not on each track, e.g., a template with 2× pitch connects with every other wire from one with 1× pitch. It is important to avoid choosing template pitches that are coprime; instead, we select pitches that have a small least common multiple.

We choose templates with varying pitches to provide choices across a range of PDN utilizations for the intermediate layers in the BEOL stack. Our templates with constant stripe width to help enable predictable obstacles for signal/clock routing [10]. Table I shows a sample template in the solution space, with fixed metal widths and pitches for the intermediate metal layers.

Table I: A PDN template for a 11-metal-layer stack in a 16nm technology.

Metal layer	Direction	Power stripe utilization	Width (μm)	Pitch (μm)
M11	V	40%	w_{11}	p_{11}
M10	H	40%	w_{10}	p_{10}
M7	V	15%	w_7	p_7
M6	H	10%	w_6	p_6
M5	V	5%	w_5	p_5

In modern designs, the top two metal layers are largely reserved for the PDN, while the supply network in the bottom two layers corresponds to a set of fixed power rails associated with the standard cells. By varying the pitches in M5–M7, a set of $|T|$ templates can be built. With three possible pitch combinations for M5–M7, we obtain $|T| = 27$. In this paper, we design templates for two foundry technologies: 16nm FinFET (16FF) and 65nm low power bulk (65LP).

B. Ranking and Pruning the Template Set

Two primary properties characterize each PDN template: quality, measured by its equivalent resistance; and utilization, measured by wire density. A denser template (with a higher wire width and lower pitch) has a lower equivalent resistance than a sparser template, but has greater congestion and may create signal/clock wiring bottlenecks. Next, we rank-order the templates to create a Pareto-optimal list.

Quality: We estimate the equivalent resistance for a case where a uniformly distributed current is drawn at the lowest-level nodes of the template. We assume that the pad locations are uniformly distributed over the chip area, and that templates are built in such a way that the pad locations are the same for all chip regions. If we simulate the injection of a unit current to pass through the pads, the computed worst-case IR drop for each template corresponds to its resistance. This resistance is used to rank-order each template in terms of their power integrity.

Utilization: The resource utilization of each template is a multidimensional vector in each layer. The relative ordering of two templates T_i and T_j in terms of utilization is not obvious if T_i is denser than T_j in some metal layers but sparser in others. To enable a linear comparison between templates, we find the fraction of resources/tracks used by each template across all layers based on the width, pitch, and track spacing of every layer in the template for a particular technology.

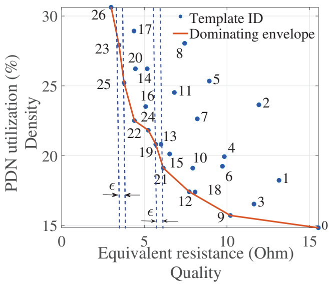


Figure 4: Template ranking based on resistance and utilization.

Fig. 4 shows the plot of utilization versus equivalent resistance for each of the 27 templates, numbered 0 through 26. Despite being denser, a few templates are of poorer quality when compared to others. This scenario occurs when a template has a higher utilization in a lower metal layer when compared to a higher metal layer. In this case, the additional stripes in the lower metal layer add to the congestion without significantly improving the quality of that template. These templates are suboptimal and are pruned from the set. The underlying cause for this suboptimality is that a designer may build the template set based on purely geometric width/pitch considerations, neglecting electrical considerations.

Proposition: Let R_i and U_i denote the equivalent resistance and utilization of template T_i; template T_i is suboptimal if there exists another template T_j such that $R_i > R_j$ and $U_i > U_j$.

A variation on the criterion is to enforce a requirement that the Pareto front must provide at least a minimum improvement in resistance per unit increase in the density, and to drop points that fail this requirement, i.e., a template whose equivalent lies within ϵ of another template that has a lower utilization, is eliminated. We refer to all dropped templates as *dominated templates*. As a result of this pruning approach the original set of 27 templates is reduced to 8 nondominated templates (i.e., 0, 9, 12, 21, 24, 22, 25, and 26).

III. Model Training

A. Generating Training Data

To generate the "golden" data that trains the neural networks (NNs), we use a simulated annealing (SA) optimizer to find an optimal power grid for a given chip configuration. We justify the use of SA due to the large discrete solution space, e.g., for $T_r = 16$ regions, with 8 templates per region, there are 8^{16} possible solutions.

The NN models are parameterized by d_c, J_c, the template set T, the package bump distribution, and region sizes. It is sufficient to characterize a small number of typical region sizes for a technology node: the trained NNs can then be used over all designs. Thus, the SA computation is a one-time cost, and it is important for it to be accurate; computational efficiency is not a significant consideration.

The SA method stochastically explores the solution space to determine a close-to-optimal solution. Its inputs are:

- The current density map of the chip
- The locations of the C4 bumps (power pads)
- The static IR drop limit, d_c
- The EM constraint (maximum wire current density, J_c)
- The congestion distribution map for signal/clock nets
- The number of regions on the chip and their size
- A pruned set of templates

We leverage the principle of locality [12] in a power grid, which states that the current paths to a node depend primarily on the density of nearby regions, to build NNs that are independent of the chip size. The use of this principle dictates the need for establishing guidelines for defining template sizes. We suggest using region sizes that are approximately equal to the VDD/VSS bump pitch. This guideline provides a lower bound to the template size ensuring the principle of locality holds, and an upper bound to obtain compelling congestion improvements. Using this guideline in our experiments, we found that the choice of an optimal template for each region depends on the region and its (up to) 8 nearest neighbor regions.[1] Thus, it is adequate to train a model based on current and congestion in 9 regions, enabling the tiling of power grid templates over a chip with an arbitrary number of regions. Additionally, this speeds up training as it reduces the dimensionality of NN input data.

The training set consists of 500 separate testcases, each corresponding to a spatial distribution of chip current and congestion. We synthetically generate these testcases by superimposing random Gaussian distributions of different mean and standard deviations to represent placement current maps. Congestion estimates for the training set are also generated randomly, maintaining a correlation between congestion and current density. For the 16FF designs, in each region, the current and congestion maps are discretized into 125×125 subregions for placement-stage CNN training. To maintain consistency between the MLP and the CNN while reflecting the limited information available at the floorplan stage, the training set for floorplan-stage MLP training averages/blurs current maps to coarser 16×16 subregions, and congestion maps to one subregion per region. We use 4×4 regions over the chip area for the 16FF designs and 10×8 regions for the larger 65LP designs that were available to us.

We then optimize these 500 testcase designs using the SA optimizer (Section III-B) that provides the optimal template for each region. Next, from this optimized data, we extract the training set for the NNs. Based on power grid locality, the template in each region is dependent on the current distribution in a 3×3 window around the current region. Therefore, for a 4×4 tessellation, we can extract 16 3×3 regions that constitute elements of the training set for the NN, including the zero-padded corner regions. Over 500 testcases, this yields $500 \times 16 = 8,000$ training set elements.

To summarize, for each of the 16 regions, each training set element at the 16FF node (65LP designs are handled similarly) consists of

- (for the floorplan-stage MLP) current distributions at the granularity of 16×16 subregions of each region, and a single congestion number for the region.
- (for the placement-stage CNN) finer-grained current and congestion distributions of 125×125, one per gcell in the layout.

Based on this information, the MLP and CNN are each trained to compute the correct output (optimized) template for the region, while incentivizing the CNN to match the MLP for similar current distributions, thus maintaining design predictability.

To train the network for each neural net, we divide the data from the golden SA optimizer into training (80% of the data), validation (10%), and test data (10%). The training data set is normalized, i.e., we subtract the mean of the data and divide by the standard deviation. This ensures that both inputs are on the same scale and neither dominates the other. The mean and standard deviation values of the training set are stored to normalize the test data during inference.

B. Optimization Problem Formulation for Generating Training Data

To generate floorplan-stage training data, the SA solver for the 500 testcases, finds a solution that optimizes, across all the regions on the chip, the utilization of the PDN, the maximum IR drop for better power integrity, and maximum current density for greater EM safety,

For placement-stage training, in addition to these constraints, the SA solver must ensure proximity of the solution to the floorplan-stage solution, in order to ensure consistency, i.e., minimal perturbation between the optimal floorplan-stage template set and the corresponding placement-stage template set. This consistency ensures that under normal floorplan-to-placement refinements, with small perturbations in the current distributions, the optimal template obtained at the floorplan stage is not greatly perturbed at the placement stage, i.e., the placement-stage solution is an incremental refinement to the floorplan-stage PDN. We use the following notations for region r:

- s_r is signal/clock congestion
- $u_{i,r}$ is the PDN utilization of the template i
- d_r is the maximum voltage drop
- J_r is average current density
- ${p^F}_{l,i,r}$ and $p_{l,i,r}$ are the pitch of layer l in template i for floorplan-stage and placement-stage optimization, respectively.
- L is the total number of layers in the PDN

The total congestion is then $c_r = s_r + u_{i,r}$, and the optimization problem at the placement-stage can be formulated as:

$$\begin{aligned} \text{Minimize: } & \textstyle\sum_{r=1}^{T_r} \left[c_r + d_{r,norm} + J_{r,norm} (+\Delta p_{r,norm}) \right] \qquad (1) \\ \text{Subject to: } & d_{r,norm} = d_r / d_c \leq 1 \\ & J_{r,norm} = J_r / J_c \leq 1 \\ & c_r \leq 1 \end{aligned}$$

[1] Regions with < 8 neighbors at the edge of the chip are zero-padded to ensure the dimensions of all the data points match, irrespective of their location.

where $\Delta p_{r,norm} = \frac{1}{L}\sum_{l=1}^{L} \frac{|p^F{}_{l,i,r} - p_{l,i,r}|}{p^F{}_{l,i,r}}$, which appears only in placement-stage optimization, is the term that minimizes the distance between the floorplan-stage and placement-stage PDNs. The terms $J_{r,norm}$ and $d_{r,norm}$ encourage denser templates while c_r encourages sparser templates. Together, these three terms encourage the optimizer to seek a balance between power integrity and PDN utilization. The normalization ensures that the magnitudes of the terms in the objective function are comparable so that no one term dominates the others. The constraints represent fundamental specifications on the PDN.

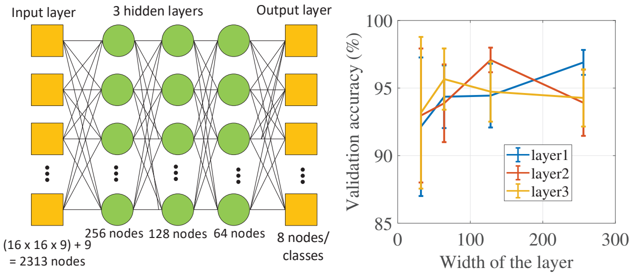
The constrained optimization problem (1) is converted into an unconstrained minimization by using the penalty function method [13]: In the cost function, the form of the penalty function is $(\alpha_i \max[0, -\text{slack}_i])$, where $i \in \{\text{congestion, IR drop, EM current density}\}$. Here, slack_i is the constraint slack; if negative, a penalty is applied. We use $\alpha_i =$ 100, 200, and 200, for congestion, IR drop limit, and EM current density, respectively, penalizing hard constraint violations on IR and EM more strongly than congestion violations, which can be mitigated by detouring wires through less congested regions.

Each step of the SA optimization involves finding the solution to the system of equations, $G\mathbf{V} = \mathbf{J}$ globally, i.e., across the entire chip. After every move, the conductance matrix, G, is incrementally updated by using the previously stored conductance matrix for each template. While this method is slow due to the cost of formulating and solving the PDN, it is a one-time characterization per technology where it is important to near-optimal solutions. We find that these solutions can be obtained using reasonable computational resources.

C. Neural Network Topologies

Next, we describe the two proposed neural networks.

MLP for floorplan-stage design: The structure of the MLP is depicted in Fig. 5(a). The input layer is of size 48×48, corresponding to a 3×3 region with 16×16 subregions for current, and 9 additional nodes representing the 9 congestion values in the 3×3 neighborhood. Therefore, the input layer has 2295 nodes. The input layer is followed by three hidden layers each of which comprises of 256, 128, and 64 nodes. Each of these fully connected layers has an associated dropout layer at the output. The dropout factor ensures the model does not overfit the training data. This factor is selected by tuning it in combination with the other hyper parameters of the MLP. The output layer has a width of 8 nodes, representing the 8 possible classes corresponding to the 8 nondominated templates.



(a) (b)

Figure 5: (a) An MLP with three hidden layers that synthesizes the optimal PDN at the floorplan-stage. (b) Hyper-parameter tuning: width of the three layers in the MLP are varied to find the combination that maximizes validation accuracy.

The size of each layer and the number of layers are the most critical hyper-parameters for the MLP. For example, Fig. 5(b) provides the mean and standard deviation in the validation accuracy when the widths of the 3 layers are varied individually while keeping the other two constant. Our grid search algorithm enumerates 64 possible solutions where the widths of the three layers can take any one of the following values: 256, 128, 64, and 32. From the plot, it can be seen that beyond certain widths, the MLP begins to overfit the training data, i.e., the mean validation accuracy decreases. Therefore, we choose a layer width of 256, 128, and 64 for the first, second, and third hidden layers, respectively.

CNN for placement-stage PDN design: The placement-stage CNN is sufficiently complex that we can use a standard CNN topology: we choose a LeNet [14] based architecture (Fig. 6). LeNet consists of two convolution layers: the first uses 32 5×5 filters, while the second uses 64 3×3 filters. The input is a 3×3 window around a region with 125×125 subregions, each with its own current and congestion attribute. Therefore, the input layer is of size $375 \times 375 \times 3$, where the three sets correspond to the placement current distributions, detailed placement congestion estimates, and the optimal template map obtained from the MLP. Each convolution layer is followed by a ReLU activation function [15], and there are two max pooling layers, which operate with zero-padding and a filter size of 2×2. The final fully-connected layer vector is of size 1024 and feeds the 8 CNN outputs, corresponding to the 8 nondominated templates. Similar to the MLP, the fully connected layer is followed by a dropout mask to prevent overfitting.

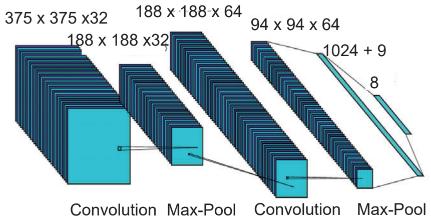


Figure 6: Modified LeNet-based CNN which is trained for PDN optimization at the placement-stage.

An ADAM optimizer [16] is used for training, which updates the learning rate during every iteration based on the second moment of the gradient. This optimizer provides faster training when compared to the classical stochastic gradient descent which uses a constant learning rate. It is important to note that it is not necessary to minimize the distance between the optimal PDN at the floorplan stage and the placement output since this is inherently captured by the fact that the training data for both NNs comes from the same SA optimizer.

We perform a grid search for hyper-parameter tuning, which involves searching through a portion of the solution space for various combinations of the hyper-parameters, to find a solution that minimizes validation error. For example, we search through a predefined solution space for various combinations of the batch size and number of epochs to provide a value that minimizes the validation accuracy of our model. Similarly, we tune the dropout factor, and the combination of momentum and learning rate of the CNN.

IV. Evaluating the ML Model

For implementation purposes, the properties of every template including its layers, width and pitch of each layer, VDD-VSS stripe spacing, sheet resistances, and via resistances are stored in a JSON format. The SA optimizer, implemented in Python 3.7, uses the JSON file and a modified nodal analysis solver to estimate the cost at every iteration by solving the system equations. All experiments are performed on a 2.20 GHz Intel® Xeon® Silver 4114 CPU.

Today, it is customary to use a static IR drop limit of 1% of V_{dd} [17], [18] in industry, and we set d_c according to this guideline in this work. Many older works on PDN synthesis (e.g., [10]) place a limit of 10% on the total IR drop, and today's tighter static IR drop limit is driven by the increased level of dynamic IR drop: standard industry flows first optimize a design for static IR at this tighter Vdd specification, which helps reduce dynamic IR drop as well. We use EM current density constraints of $J_c = 3\text{MA/cm}^2$ at 16FF and $J_c = 4.8\text{MA/cm}^2$ [19] at 65LP.

We measure the improvement in resource utilization, based on the widely-used ACE metric [20], which estimates the improvement in congestion only if the region is critical, i.e., if it has an average signal congestion value greater than a certain threshold, set here to 50%. The metric thus determines where the saved PDN resources can be potentially utilized for signal/clock routes.

Since the SA algorithm must analyze a PDN with over 1 million nodes at each iteration, it takes around 20 minutes to converge to a near-optimal solution for our each of testcases and a fixed template set. To obtain our complete data set with 500 testcases (with 16 regions each, a total training set size of $500 \times 16 = 8000$) we execute over 30 processes in parallel. The MLP and CNN are both implemented using TensorFlow and trained on the same system described above. The MLP takes 24s to train, while the CNN takes 178 minutes. It is important to note that both the training data generation and the training itself are one-time non-recurring costs for a specific technology and specific region size, and therefore their overhead is worthwhile as it delivers fast near-optimal, safe-by-construction PDN synthesis for any design with no human in the loop. We train the neural networks based on the parameters of foundry 16FF and 65LP technologies.

A. Inference Scheme

Each test design provides a full-chip current distribution and congestion map as shown in Fig. 8(a). For each region on the chip, we select the template (chosen to have the same area as in the training set), by taking the 3×3 neighborhood of the region extracting the current distribution and congestion distribution according to the subregion granularity (16×16 for current and a single number for congestion at the floorplanning stage; 125×125 [175×175] for current and congestion at the placement stage for 16FF [65LP]). Finally, for each testcase, we take one region and its 3×3 window at a time, and input the normalized current and the congestion values (by subtracting the mean and dividing by the standard deviation of the training set) into the MLP or CNN, which selects the optimal PDN template ID.

B. Validating the Neural Network Classification

1) Validation on synthetic testcases: As stated earlier, 10% of the generated data points in Section III-A are used to test the results of training. The confusion matrix which depicts the classification accuracy for the test set is shown for both the MLP and CNN in Fig. 7(a) and (b), respectively. In each matrix, the classes are sorted in the decreasing order of their equivalent resistance. Therefore, any misclassification which lies in the lower triangle of the confusion matrix is still IR-safe. Hence, for the MLP we get a 98.875% IR-safety guarantee and for the CNN we get a 97% IR-safety guarantee. It is important to note that the confusion matrix is a conservative predictor of the accuracy of our overall PDN synthesis scheme. This matrix represents the accuracy for the template ID for *only one* region of the chip, considering a 3×3 window around the region. It is likely that if one template is optimistically chosen, the templates of the regions around it will be conservatively chosen (as seen from the confusion matrix, a vast majority of template choices are pessimistic). This observation is borne out across our testcases and results in Table II, where all our synthesized grids are IR-safe and EM-safe.

2) Validation on real testcases: Next, we validate each of the models on current distributions obtained from real designs. The evaluated designs include four different RISC cores, labeled Core1 [11] with 171,697 cells and 10 macros, Core2 [21] with 209,516 cells and 26 macros, Core3 [22] with 56,954 cells and 4 macros, and Core4 [23] with 365,075 cells and 36 macros. The first core is implemented in 16FF and the last three in 65LP technology.

Fig. 8(a) shows the input current distribution and region-wise congestion of Core1 at the floorplan stage. The data in this figure is fed into the trained MLP to synthesize an optimal PDN (Fig. 8(b)). The PDN is IR and EM-safe with a worst-case IR drop of 5.58mV and maximum current density of 2.74MA/cm^2.

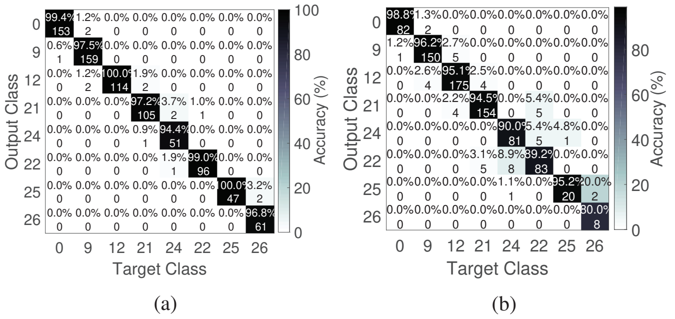


Figure 7: Confusion matrix for (a) MLP classifier with 800 floorplan testcases and (b) CNN classifier with 800 placement testcases.

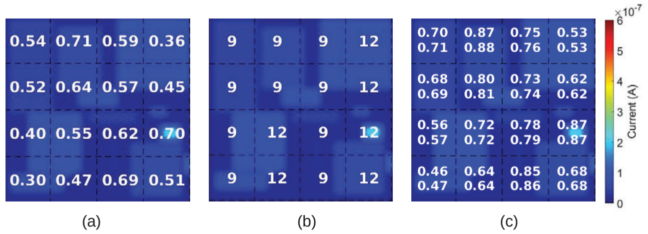


Figure 8: ML-based demonstration of the inference scheme at the *floorplan stage* of Core1 [11] designed using a foundry 16nm FinFET technology: (a) coarse current and average congestion distributions; (b) synthesized templates in every region using MLP; and (c) average congestion (signal + PDN) in every region after classification of different templates (top number); average congestion (signal + PDN) in every region after using a uniform PDN across the entire die (bottom number).

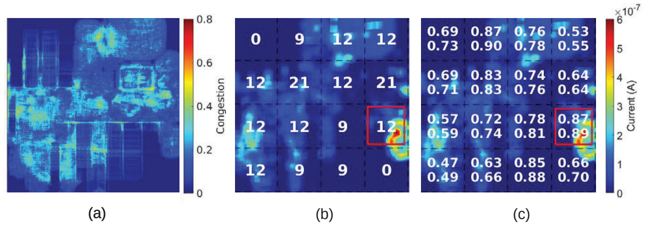


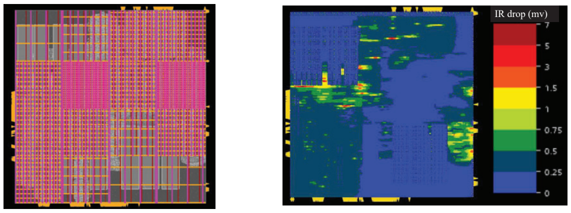
Figure 9: CNN-based demonstration of the inference scheme at the *placement stage* of Core1 [11]: (a) input signal congestion map, (b) current distribution and the synthesized template-based PDN, (c) average congestion (signal + PDN) in every region after classification of different templates (top number); average congestion (signal + PDN) in every region after using a uniform PDN across the entire die (bottom number).

For comparison, we optimize the power network by imposing a uniform grid over the entire chip, enumerating the 8 choices of the templates. The number at the bottom of each region in Fig. 8(c) is the predicted congestion if a uniform grid which just meets IR drop is used, while that at the top is the template-based total congestion. For Core1, as against the uniform PDN (template 21) that satisfies the static IR drop limit of 8mV, our floorplan-stage PDN provides an ACE metric congestion improvement of 1.47% (1,188 tracks).

For placement-stage optimization, we use the CNN to build a PDN based on the post-placement congestion (Fig. 9(a)) and current (Fig. 9(b)) distribution for Core1. The CNN-synthesized output for Core1 is depicted by the template IDs shown in Fig. 9(b), and it can be verified to be a small perturbation to the floorplan-stage PDN in Fig. 8(b), i.e., the templates change at most to the next denser template. This refinement ensures the PDN still meets IR and EM constraints at placement, and also improves the ACE metric for congestion by 2.39%

Table II: Performance evaluation of both the neural networks on different RISC V cores implemented in 16FF and 65LP technologies.

Testcase	Static IR limit	Tech. node	Floorplan						Placement					
			Uniform grid		MLP-synthesized				Uniform grid		CNN-synthesized			
			Max IR drop	EM: Worst-case $J_{r,norm}$	Max IR drop	EM: Worst-case $J_{r,norm}$	Congestion improvement (ACE metric)	Average number of tracks saved in critical regions	Max IR drop	EM: Worst-case $J_{r,norm}$	Max IR drop	EM: Worst-case $J_{r,norm}$	Congestion improvement (ACE metric)	Average number of tracks saved in critical regions
Core1 [11]	8mV	16FF	5.67mV	90.45%	5.58mV	91.55%	1.47%	1,188	6.72mV	91.72%	6.74mV	93.55%	2.39%	1,360
Core2 [21]	12mV	65LP	11.56mV	96.22%	11.66mV	98.57%	2.36%	1,278	11.75mV	96.39%	11.89mV	98.71%	3.22%	2,148
Core3 [22]	12mV	65LP	10.21mV	93.66%	10.81mV	95.84%	1.91%	768	8.47mV	91.11%	10.98mV	94.13%	3.02%	1,224
Core4 [23]	12mV	65LP	11.44mV	95.32%	11.73mV	96.50%	2.66%	2,278	11.58mV	96.94%	11.82mV	98.24%	3.31%	2,574
IBM [24]	8mV	16FF	5.05mV	93.61%	5.60mV	95.18%	2.12%	1,348	–	–	–	–	–	–



(a) (b)

Figure 10: (a) The synthesized template-based PDN (showing layers M5-M7 only) in the 16 regions of Core1, and (b) its corresponding IR drop map obtained from Voltus [25].

(1,360 tracks) when compared to a uniform grid (Fig. 9(c)). The red rectangle in Fig. 9 highlights a region with a high current density and high signal congestion. In this region, the CNN selects a sparser template (12) and compensates for the high current by selecting a denser template (21) in a neighboring less-congested region.

The inference scheme for Core1 has a total run-time of 1.3s for the MLP, while the inference scheme on the CNN has a total run-time of 20.2s, which includes 15.3s to load the TensorFlow database. The results for Core2, Core3, and Core4 core are summarized in Table II, which lists the congestion improvement, worst-case IR drop, and the worst-case normalized current density ($J_{r,norm}$), defined in (1), at both placement and floorplan stages. It is worth mentioning that a 1–3% improvement in the ACE metric for congestion is significant for two reasons: (i) this percentage improvement releases thousands of tracks (Table II), and (ii) by the nature of the ACE metric, which measures the congestion in only regions which are critical (signal congestion greater than 0.5), the released tracks have a high potential to aid design closure. The final testcase corresponds to the widely used IBM benchmarks [24]: although there are six such benchmarks, all have the same underlying current distributions, which are available at the floorplan level. Since placement-stage current distributions are not available, we can only apply the MLP to synthesize its PDN. The benchmark set does not provide congestion information, but we generate this using current-congestion correlation statistics.

3) Validation with a commercial flow: We validate our methodology using Innvous [26] on Core1 in 16FF technology. We begin at the floorplan stage where an initial uniform PDN is synthesized across the chip. After placement, based on the locations and power drawn by each cell, the current distributions are generated. An early global route provides congestion (signal and PDN) estimates. The three inputs (i) current map, (ii) signal congestion estimates, and (iii) template IDs from the floorplan stage, are fed into the trained CNN.

We use the inference scheme (Section IV-A) to infer the optimal template in every region. This final synthesized template-based PDN is shown in Fig. 10(a). The IR drop map of the design obtained using Voltus [25] for the predicted template-based PDN is shown in Fig. 10(b), verifying that the PDN is IR-safe (8mV limit).

V. Conclusion

This paper addresses the iterative and time consuming nature of a PDN synthesis and optimization by using a two-stage neural network approach to synthesize a IR- and EM-safe optimal PDN. The one-time cost involved in training the models is compensated for when an optimized PDN can be rapidly be synthesized for several designs. On average we save about 1,850 tracks (3% congestion relief) in the congestion-critical regions. These saved resources can potentially be vital to aid timing closure in high performance chips.

References

[1] H. Su, *et al.*, "Congestion-driven codesign of power and signal networks," in *Proc. DAC*, pp. 64–69, 2002.

[2] J. Kozhaya, *et al.*, "Multigrid-like technique for power grid analysis," in *Proc. DAC*, pp. 480–487, 2001.

[3] M. Zhao *et al.*, "Hierarchical analysis of power distribution networks," in *Proc. DAC*, pp. 150–155, 2000.

[4] P. Li, "Power grid simulation via efficient sampling-based sensitivity analysis and hierarchical symbolic relaxation," in *Proc. DAC*, p. 669, 2005.

[5] H. Zhuang, *et al.*, "Simulation algorithms with exponential integration for time-domain analysis of large-scale power delivery networks," *IEEE T. Comput. Aid D.*, vol. 35, pp. 1681–1694, Oct. 2016.

[6] X.-D. Tan *et al.*, "Reliability-constrained area optimization of VLSI power ground networks via sequence of linear programmings," in *Proc. DAC*, pp. 78–83, 1999.

[7] X. Wu *et al.*, "Area minimization of power distribution network using efficient nonlinear programming techniques," *IEEE T. Comput. Aid D.*, vol. 23, pp. 1086–1094, July 2004.

[8] T. Tseng, *et al.*, "A power delivery network (PDN) engineering change order (ECO) approach for repairing IR-drop failures after the routing stage," in *Proc. VLSI-DAT*, 2014.

[9] W.-H. Chang, *et al.*, "Generating routing-driven power distribution networks with machine-learning technique," in *Proc. ISPD*, pp. 145–152, 2016.

[10] J. Singh and S. S. Sapatnekar, "A partition-based algorithm for power grid design using locality," *IEEE T. Comput. Aid D.*, vol. 25, pp. 664–677, Apr. 2006.

[11] "OpenCelerity: Rocket core." http://opencelerity.org.

[12] E. Chiprout, "Fast flip-chip power grid analysis via locality and grid shells," in *Proc. ICCAD*, pp. 485–488, 2004.

[13] J. A. Snyman, *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Springer Boston, MA, 2005.

[14] Y. LeCun, *et al.*, "Gradient-based learning applied to document recognition," *Proc. of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[15] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICICML*, pp. 807–814, 2010.

[16] D. P. Kingma and J. Ba, "ADAM: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[17] M. Patil, *et al.*, "A multi-perspective approach to IC power grid development for 7nm based designs," in *ACM/IEEE Design Automation Conference – Designer Track*, 2019. (slides available at dac.com).

[18] A. B. Kahng, *et al.*, "Power delivery pathfinding for emerging die-to-wafer integration technology," in *Proc. DATE*, pp. 842–847, March 2019.

[19] J. Lienig, "Electromigration and its impact on physical design in future technologies," in *Proc. ISPD*, pp. 33–40, 2013.

[20] Y. Wei, *et al.*, "GLARE: Global and local wiring aware routability evaluation," in *Proc. DAC*, pp. 768–773, 2012.

[21] "Black Parrot." https://github.com/black-parrot.

[22] "OpenCelerity: Vanilla core." http://opencelerity.org.

[23] "Ariane." https://pulp-platform.github.io/ariane/docs/home.

[24] S. R. Nassif, "Power grid analysis benchmarks," in *Proc. ASP-DAC*, pp. 376–381, March 2008.

[25] "Cadence Voltus IC power integrity solution user guide." http://www.cadence.com.

[26] "Cadence Innovus user guide." http://www.cadence.com.