2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION JAPAN

Machine Learning-based Smart Assessment of User Floorplan Quality without running Place & Route

Harn Hua Ng, Plunify, Singapore (*harnhua@plunify.com*)
Kirvy Teo, Plunify, Singapore (*kirvy@plunify.com*)

***Abstract*—A new approach to evaluate chip design floorplans by training an Artificial Intelligence model.**

Keywords—ASIC; FPGA; Floorplanning; Machine Learning;

I. Problem Statement and ML-Based Solution

Floorplanning is a process that uses physical constraints (region constraints and assignment of design blocks to these regions as well as IO assignments) to guide and drive the "placement" process to achieve higher performance. In many cases, floorplan constraints backfire and lead to unexpected placement failures, routing failures, or timing violations with different degrees of severity. Most of the routing failures or struggles are induced by the floorplan that creates artificial local or global congestion.

Moreover, when placement and routing failures occur, iterative manual changes of the floorplan are tedious, non-deterministic, and excessively time- and compute resource-consuming. The designer's frustration grows significantly with the number of iterations and lack of identifying and/or understanding of the root causes of these failures.

In this contribution, a Machine-Learning based approach is proposed to tackle these issues without the need to even run the placement or routing tools. The insight compiled by this innovative approach allows designers to easily get an understanding of the quality of the floorplan that could lead to placement or routing failures, or severe timing violations. This allows designers to

- explore various floorplans within a short time (no need to run time- and resource-hungry place and route tools),
- deterministically predict the performance outcome once a floorplan is deemed "good" or even "good enough" by the ML-based assessment method.

This shift in methodology does not eliminate the need to compile a design, but allows users to screen out severely flawed floorplans. In doing so, designers potentially save weeks of trial-and-error, and are able to achieve timing convergence in short order once the floorplan has been fixed and deemed "good". The deterministic approach has removed frustration and stress faced by the design teams when they tackled the floorplan issues manually, and has allowed management to keep schedules on track, and save money.

II. Methodology and Approach

The approach being described uses Machine Learning (ML) and Artificial Intelligence (AI) technologies to speed up floorplanning and reduce compilation turnaround time in both ASIC and FPGA design. A mix of design heuristics and Deep Learning architectures is employed to generate new floorplans and improve the quality of Place and Route. Based on experiments with commercial designs, this method is able to predict the final timing performance in terms of WNS with up to 80% accuracy. Furthermore, design teams using this approach can reduce the turnaround time for an end-to-end solution by 5x. Figure 1 shows the intended use-case.

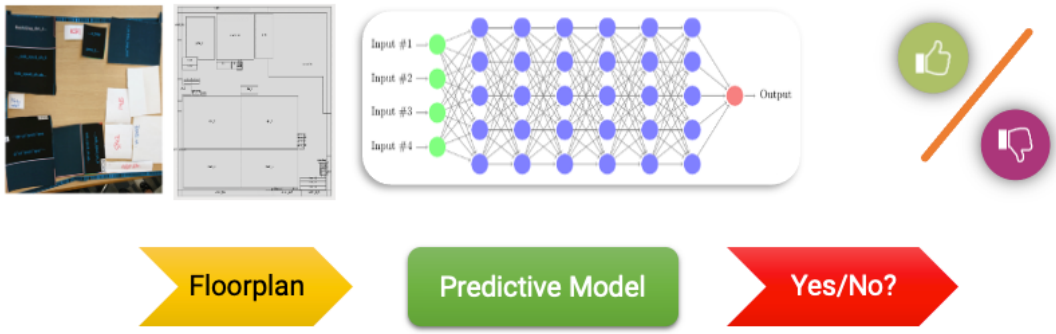

Figure 1: Intended use-case where a trained AI model predicts floorplan QoR before deciding to compile the design.

A. *Objectives*

1. Develop an AI model that points out "good" and "bad" floorplans without actually running Place & Route.
2. Generate floorplans and use the developed AI model to select good ones for production.

B. *Output Metric: Heatmap of Worst Timing Slack (WNS) Per Cell*

The AI model, when fully trained, plots the congestion or WNS of each cell in the design and composes them into a heatmap like the one below. Visually, a heatmap is more intuitive for end-users to understand why one floorplan is ranked higher than another. A sample WNS heatmap is shown in Figure 2.

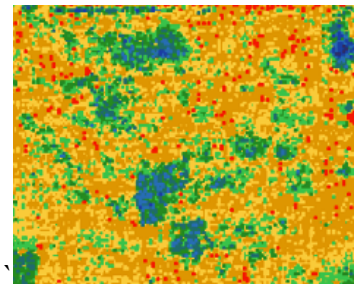

Figure 2: Sample Heatmap showing a scale of WNS values ranging from severe (red) to acceptable (green)

The end-user then choose the floorplans with the fewest red spots (predicted to produce better timing performance) and prioritizes them for actual placement and routing. Bad floorplans are discarded to save time and compute resources.

III. Development Environment

Python-, Tcl- and TensorFlow-based libraries in a Linux operating system environment are used for development, model training and prediction.

Five (5) commercial ASIC and FPGA designs from customers and industry partners are used for this effort. On average, commercial chip design tools take 5~8 hours to place and 12~24 hours to route each of these designs.

Specific tasks:

1. Determine the types of input data (“features”) based on domain knowledge. that influence the target performance metrics.
2. Format the input data into the required formats required by the AI model.
3. Architect a custom Machine Learning model to train the input data
4. Generate predictions of the performance metrics as “heatmap” images.

5. Evaluate prediction quality through visual inspection of the heatmap images and output metrics.

During a manual floorplanning process, experienced engineers evaluate about 10 design characteristics to create new floorplans and estimate their performance results. This project augments the human experience by analyzing and learning from the raw data generated by the chip design tools.

IV. Workflow Design

As ASIC and FPGA design flows share similar characteristics, the following frameworks are created to support both flows. Figures 3 and 4 show the workflows for ASIC and FPGA design respectively.

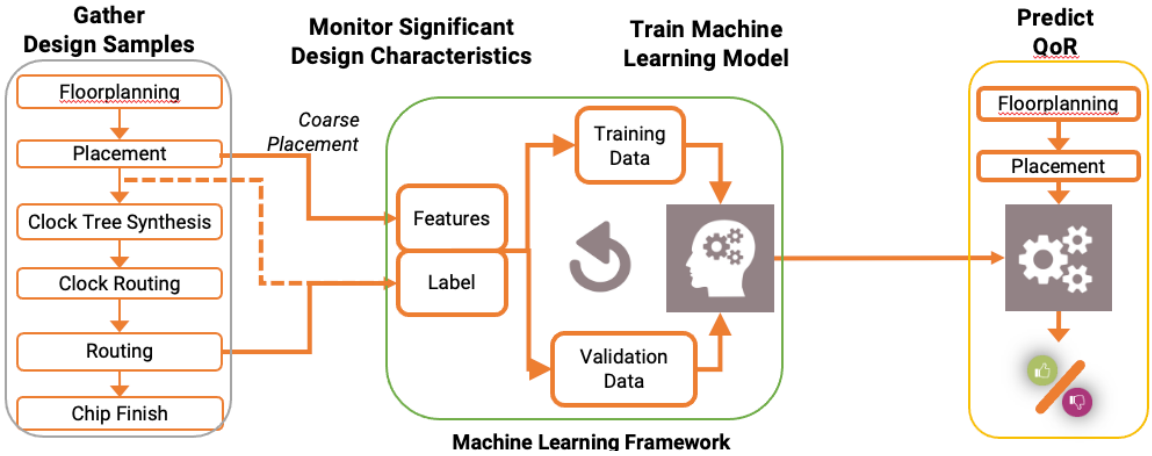

Figure 3: ASIC workflow for extracting relevant design characteristics to train and use an AI model.

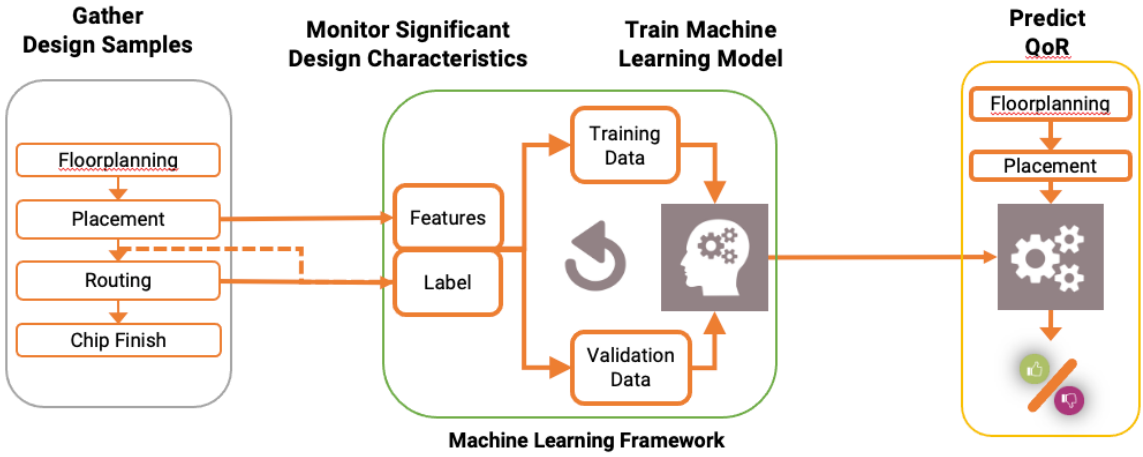

Figure 4: FPGA workflow for extracting relevant design characteristics to train and use an AI model.

Each workflow can be split into three parts; Section IV.A "Determine Features and Extract Input Training Data" describes the design characteristics that are extracted and utilized as training data. Section IV.B "Generate Floorplans and Obtain Training Labels" explains the process of generating new floorplans and running coarse/quick placement attempts to get WNS estimates for evaluation. Finally, the AI model is trained and fine-tuned in Section IV.C "Train and Fine-Tune Model".

A. Determine Features and Extract Input Training Data

The following design characteristics comprise the set of input features used to train our model. As they are approximations and abstractions that the Machine Learning framework use to understand a design, these features will vary according to different chip architectures and domain knowledge. Therefore, this set is not comprehensive

and should be expanded upon in future work. It is worth noting that training runtime is expected to increase along with the number of features as well.

1) Design as an XY Grid

Firstly, a chip design is modelled as a 2-dimensional grid with each pair of Cartesian coordinates representing one or more placement locations. This allows chip designs of different sizes, aspect ratios and resolutions to be used for training. Figure 5 illustrates this concept of overlaying circuit elements on a grid.

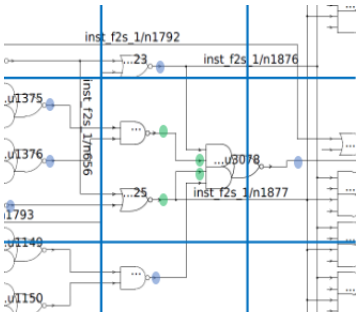

Figure 5: Modelling a chip design as a 2D grid

2) Cell Density

Next, depending on how each grid location is defined, this feature tracks the number of combinational or sequential elements in a single XY location. A larger cell density means a greater likelihood of congestion.

3) Pin Density

Similar to 2.1.2 Cell Density, except that this feature counts the number of pins.

4) External Connectivity

This is the number of connections between cells within a single XY location to those in other locations. Connectivity affects placement and routing congestion.

5) Internal Connectivity

If there are multiple cells within an XY location, this characteristic gives the number of connections between them.

6) Macro Regions

These are portions of the chip that are occupied by specialized circuitry, for example, memory or DSPs, and are typically bigger than cells or registers. Macros sometimes have specific location requirements so the place and route tool has to work around them. This feature keeps track of such specialized locations.

7) Occupied Regions

The initial placement has already assigned logic elements to specific XY coordinates. Areas of the chip that have been placed are differentiated from unoccupied regions, so that the model knows how much "freedom" it has to modify the placement.

The above features form the "feature map" of each floorplan that is used to train our model.

B. Generate Floorplans and Obtain Training Labels

The designs used for this work have been compiled once. Some of them failed routing but all have placement ("initial placement") data. Based on the following heuristics, eight (8) new floorplans are generated via scripts for each initial placement to form a total of 40 new floorplans.

- Keep densely-connected cell instances in centralized locations.
- Increasing the spacing between cell instances in congested regions.
- Move only instances with severe negative slack.

- Obey design rules and user IO constraints with respect to how certain resource types or instances can or cannot be moved.

Subsequently, coarse/quick placement is run with the new floorplans to extract more training data and their resulting WNS. The post-placement WNS is used to grade each new floorplan as “good” or “bad”, and teaches the AI model how to evaluate each floorplan.

Furthermore, with each floorplan, the WNS estimates of critical instances are also monitored and used as training labels. In Figure 6, each column is a different floorplan and each row shows the approximated WNS of an instance. Red represents highly-negative slack and green is used for positive slack.

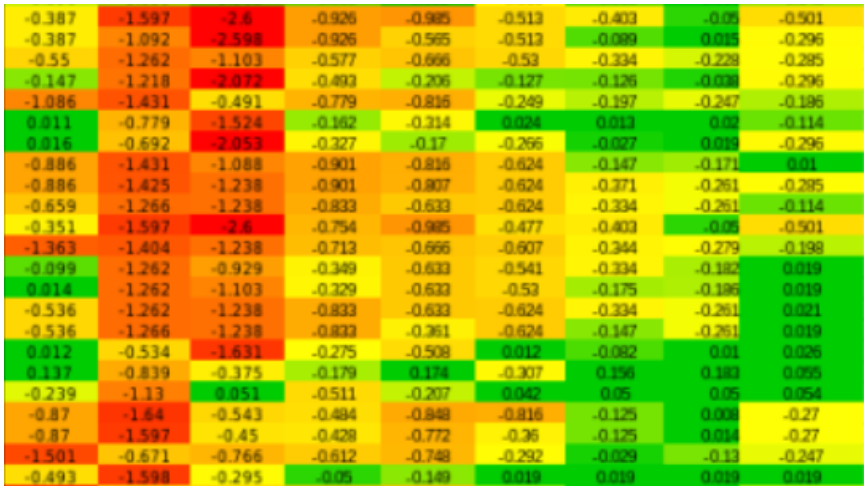

Figure 6: Post-placement, pre-routing WNS of design instances

By correlating these training labels with the input features, the model learns how to evaluate if a floorplan is good or poor.

C. Train and Fine-Tune AI Model

The Machine Learning model is created based on the structure of a Convolutional Neural Network (CNN). CNNs are commonly used to train on images such as those of animals, plants or objects. As chip design data is scarce and of a different nature than image data, a custom CNN model is designed from scratch and fine-tuned the structure via experimentation.

The CNN is a simple “two-class” one that identifies if a floorplan will improve the final WNS.

- Class 1: WNS does not improve.
- Class 2: WNS improves.

If the predicted possibility of Class 2 is higher than that of Class 1, the model will classify the proposed floorplan as improved WNS and suggest that the end-user adopt it.

During training, the CNN outputs a predicted WNS value, which is compared with the actual WNS to see how accurate the prediction is. To make it easier for the end-user to inspect and evaluate floorplans, heatmaps are generated as well.

The number of training iterations can affect the AI model quality. The images in Figure 7 show differences in the predictions for training iterations of 100 (left) and 1000 (right) respectively.

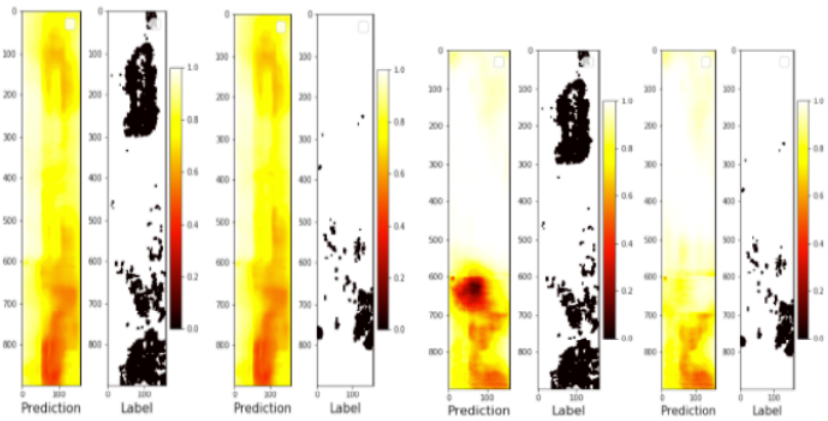

Figure 7: (Left) 100 Training Iterations; (Right) 1000 Training Iterations

Each pair of "Prediction" and "Label" images represents the predicted WNS and actual WNS of a single floorplan. There are two floorplans in each column above.

- Prediction: Heatmap of predicted WNS where red is the most negative value and white is the most positive value.
- Label: Actual WNS where black represents negative values.

When the number of training iterations is small, more regions of the chip design are incorrectly predicted to have negative WNS.

Standard statistical metrics such as training and validation loss (total amount of errors) and training and validation accuracy (percentage of correct predictions) are employed to evaluate the trained model. To further improve the model's metrics, different statistical methods for loss functions and optimizers are applied. In addition, Machine Learning variables called hyper-parameters that control the learning process are adjusted.

V. RESULTS

On average, the tuned model is able to predict instances with negative WNS with 80% accuracy.

Figure 8 shows the predictions and actual WNS heatmaps of four (4) floorplans from a sample of 1,000 floorplans generated for the five commercial designs. Based on the heatmaps, Floorplan 3 has the least amount of predicted negative WNS and thus should be the first floorplan to run through place and route.

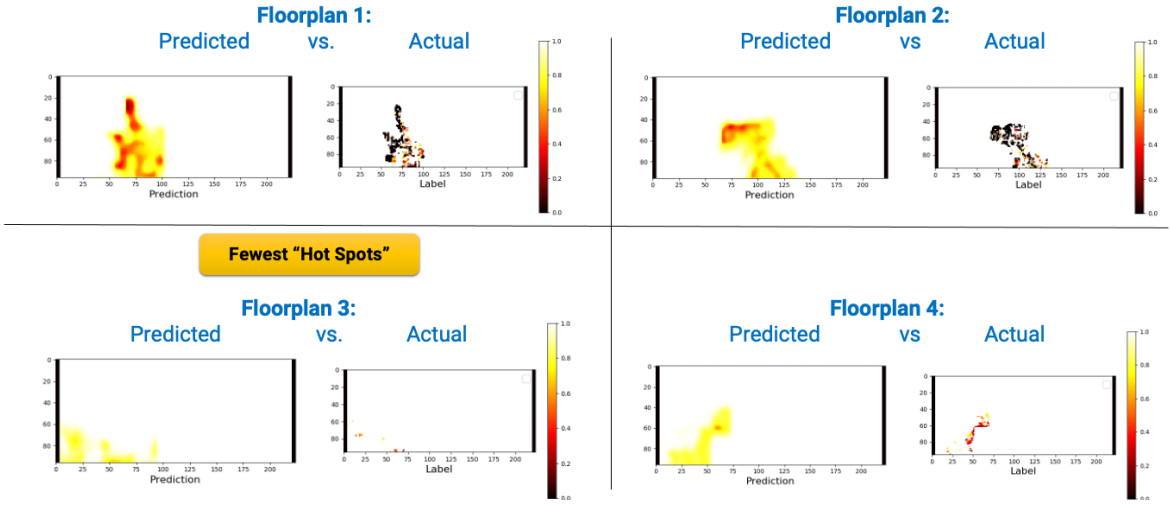

Figure 8: Evaluating 4 Floorplans - WNS Heatmap Predictions vs. Actual

Out of close to 1,000 floorplans that were generated via automation, about 50% yielded actual WNS improvements after placement and routing.

Assuming that placement runtime is half of routing runtime (N) and that it takes three (3) attempts on average for the trained AI model to find a floorplan that improves WNS, this method accelerates turnaround time by close to 5x. Table 1 contains calculations for the turnaround time to create and verify new floorplans, according to the experience level of the engineers and the performance of the trained AI model.

Table I. Turnaround Time to Create and Verify Floorplans

Categorized according to experience levels			
	Senior Engineer	***Junior Engineer***	***Trained AI Model***
Routing Runtime	N		
Placement Runtime	0.5N		
Average Number of Floorplans Needed to Improve WNS	3	8	3
Turnaround Time	(0.5 + 1) * N * 3 = 4.5N	(0.5 + 1) * N * 8 = 12N	0.5N * 3 + N = 4.5N

In other words, if existing workflows require one month to derive a better floorplan, this approach can achieve the same in less than a week.

VI. Lessons Learnt and Future Developments

In chip design flows, poor performance results are often discarded in favor of newer attempts at improving the end-metrics like timing and area. However, the design characteristics of failures can be used to train AI models instead. Formal dataflows should be defined and adopted to extract training data from compilations that fail performance targets.

Not all the design characteristics that affect performance are well-understood. Beyond known factors like logic level or fanout, there are other 1st- and Nth-order variables that can be extracted and used as training features. This is a ripe area for further research and development.

As AI models improve, so do chip architectures and process technologies. Hence, the selected features should be re-assessed on a regular basis to see if they are still effective. AI models trained with the specific features and algorithms developed in this project can be combined with other AI models to become better in predicting the quality of results in ASIC and FPGA design.

References

[1] B. V. Do, K. W. Ng, H. H. Ng, K. Teo and S. Y. Yuan, "Predicting Timing Bottlenecks in Place & Route using Machine Learning," Design Automation Conference, San Francisco, December 2021.

[2] D. Maarouf et al., "Machine-Learning Based Congestion Estimation for Modern FPGAs," 2018 28th International Conference on Field Programmable Logic and Applications (FPL), Dublin, 2018, pp. 427- 4277, (*doi: 10.1109/FPL.2018.00079*).

[3] C. Pui, G. Chen, Y. Ma, E. F. Y. Young and B. Yu, "Clock-aware ultrascale FPGA placement with machine learning routability prediction: (Invited paper)," 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Irvine, CA, 2017, pp. 929-936, (*doi: 10.1109/ICCAD.2017.8203880*).

[4] W. Li, S. Dhar, D. Z. Pan, "UTPlaceF: A Routability-Driven FPGA Placer with Physical and Congestion Aware Packing," 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Austin, TX, 2016, (*doi: 10.1145/2966986.2980083*).

[5] K. E. Murray, V. Betz, "HETRIS: Adaptive floorplanning for heterogeneous FPGAs," International Conference on Field Programmable Technology (FPT), Queenstown, New Zealand, 2015, (*doi: 10.1109/FPT.2015.7393136*).