Early Stage Integrated Circuit Design Efficacy Prediction using Congestion and Cell Density Maps

Isuru Daulagala
NVIDIA Corporation
Santa Clara, CA
idaulagala@nvidia.com

Praveena Kode
NVIDIA Corporation
Santa Clara, CA
pkode@nvidia.com

Abstract

The Physical Design flow of integrated circuit design which is also known as the backend flow consists of converting a netlist or connectivity information to geometric representations of wires, transistors and other manufacturing material. This flow is partitioned into various substages such as floorplanning, placement, clock tree synthesis and routing. At each stage designers are tasked with finding the optimal design based on many constraints such as power, performance and area while performing often executing NP-complete algorithms to complete each substage. The efficacy of a final design is often unintuitive at early stages. We used congestion and cell density images from each step to predict the final design efficacy in terms of the number of Design Rule Constraint (DRC) violations of the final design. We used popular Convolutional Net Architectures such as Resnet, VGG as well as RNNs to sequentially take glimpses of congestion and density map images to classify the final DRC numbers. The architectures were able to classify the final DRC numbers with an accuracy of 57%, 71%, 79% and 87% for the placement, clock tree synthesis, route and post-route stages respectively.

1. Introduction

The Physical Design flow of integrated circuit design which is also known as the backend flow consists of converting a netlist or connectivity information to geometric representations of wires, transistors and other manufacturing material. A summary of the flow is presented in Figure 1 . The physical Design flow itself can be divided to the following main substages :

1. **Floorplanning** - Memory elements/RAMs and other third party IP (Intellectual property) blocks are placed in the circuit area
2. **Placement** - Transistors are placed based on connectivity information
3. **Clock Tree Synthesis** - The clock signal is routed to all flip flops
4. **Routing** - All other signal wires are connected/routed

The algorithms behind majority of these substages[21][10] are NP-complete and have a large runtime as seen in Figure 2. The Physical Design Engineer runs tools which execute such algorithms while fine-tuning certain parameters which are available to him or her. An execution path based on a list of values for these parameters for which the tool runs algorithms for multiple stages described above is known as a **run**. The efficacy of the final design is often measured by two metrics - if designs meet their timing targets and the amount of Design Rule Checking (DRC) violations there are in the final design. DRC rules are speficied by the foundary fabricating the chip. An example of a DRC violation is two wires being too close. If there are violations in a design, the chip cannot be fabricated. At the end of the Physical Design flow, the Physical Design often spends a large amount of time fixing DRCs manually. If the Physical Designer cannot fix all of

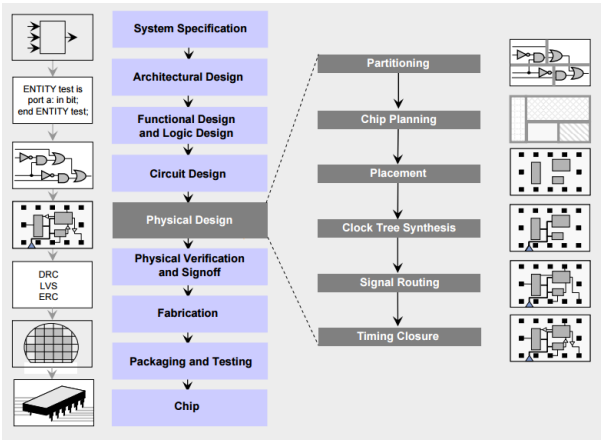


Figure 1: The Physical Design Flow as part of the overall Intergrated Design process [17]

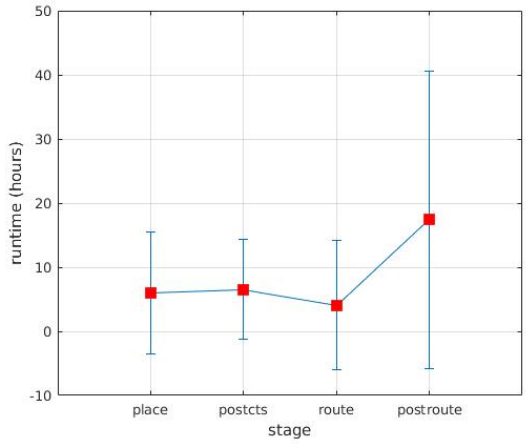


Figure 2: The mean and standard deviation for the runtime for substages in Physical Design for a Mobile SoC

the DRCs or if there are too many of them, the Physical Designer will have to launch a new *run* with different parameters. As shown in Figure 2, this whole process from placement to postroute stage can take more than 40 hours in total. So it is necessary to gain insight about a run at the beginning of the Physical Design flow to save engineering effort as well as computational resources.

In order to determine if a specific run has too many DRCs we consider two classes of images

1. **Congestion Maps** - Congestion maps describe the space of a design in terms of congestion of metal layers or wires. The bins or colors are directly proportional to the amount of metal layers/wires in a given unit volume. Since the metal layers/wires can span in 3-D space, a unit volums is considered.
2. **Cell Density Maps** - Cell Density Maps describe the space of a design in terms of the number of transistors or cell in a given area. The bins or colors are directly proportional to the amount of cells in a unit area.

The changes to the Congestion and Cell Density Maps as the Physical Design flow progresses is shown in Figure 3.

This project aims to :

1. Predict the number of DRC violations in a given run at different substages considering congestion and cell density maps
2. Predict if a run needs to be re-executed because it will result in too many DRC violations

2. Related Work

The amount of work to predict the efficacy of a run at an early stage is surpisingly scarce. Hence, the amount of literature that try to predict the number of DRC violations is even scarcer. The existing literature can be divided into three categories : 1) literature on predicting the number of DRC violations at an early stage 2)literature on prediting congestion at an early stage and 2) literature on predicting the efficacy of a run that is not related to DRC violations at an early stage.

Chan et. al[6]'s work is the closest to this work. They tried to extract number of features off at the placement stage. They use a somewhat small threshold of 50 DRCs to classify if a design ends up being routable. Although they achieve a somewhat high accuracy (67%), their F1 scores are extremely low (21%) owing to their data skewed to being unroutable. They also use designs with older feature sizes (28nm FDSOI). With newer designs and lower feature sizes, the design rules are much more complicated. [7] tried to predict hotspots (regions with DRC violations) based on features extracted during the routing stage. Both these publications used Support Vector Machines based pre-selected features without actually using the congestion maps.

The congestion maps generated at the placement stage are not based on real congestion of metal layers since the design hasn't been routed yet. Majority of work[4][15][24][5][25][20][18][19][13][12][23] in the past has gone to estimating the congestion at an early stage. These work has helped generate congestion maps which are being used in this project. Yet, existing work apart from two papers deccribed above do not tackle predicting the number of DRCs directly.

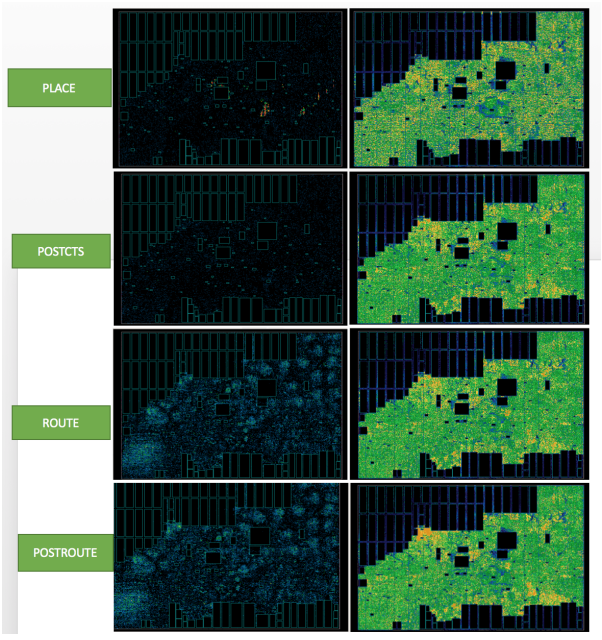


Figure 3: The changes in Congestion and Cell Density maps with the progression of the Physical Design Flow

Jiao and Daulagala [16], tried to predict if a run would fail based on the logfile that is generated by the tool during the run. They tried to generate a language model to predict failures by training logfiles. While they were able to predict if a specific run could fail based on a logfile, these failures were not necessarily only due to the underlying algorithms in the stages not being able to converge. For example, if there are libraries that are needed to execute the run that were missing, the run would fail.

None of the work discussed above use congestion map images as features directly. To the best of our knowledge, we are the first work to use visual recognition techniques and deep neural nets to predict the number of DRC violations.

The use of machine learning in Physical Design is a relatively new research area [26]. Majority of the research improving the algorithms of Physical Design have come from 3rd party Electronic Design Automation (EDA) vendor companies who don't have access to the data the companies which actually design the circuits.

3. Methods

The problem of predicting the number of DRCs based on the congestion and cell denisty maps of different stages was converted into a classification problem by binning the number of DRCs to six classes as described in the Dataset section. We considered 3 architectures which are described below. The first 2 are winning architectures of the ILSVRC challenge[3].

3.1. VGG

The original VGG Network[22] composed of blocks of two convolutional layers and max pool layers stacked together with ReLU activation units in the convolutional layers. In the VGG-19 Network there were 5 such blocks (15 layers), followed by 3 fully connected layer and the softmax layer. 3x3 convolutional filters were used with a stride of 1. 3x3 kernels were used since stacked convolutional layers would result in a large receptive field while keeping the benefits of a smaller kernel such as parameter such. Since this paper was published before the advents of batch normalization [14], batch normalization wasn't used in the original paper. However, for this project we added a batch normalization layer creating a [conv-conv-maxpool-batchnorm] block.

3.2. Resnet

Residual Networks of Resnets are composed of residual blocks shown in Figure **??**. Resnets are revolutionary since during backpropagation, it allows gradients to pass directly through block skipping what's in the block. Multiple residual blocks can be stacked. We used the implementation described shown in Figure **??**

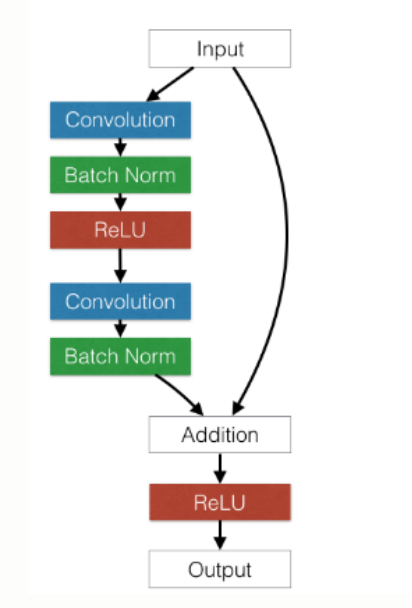


Figure 4: Residual Block Architecture [9]

3.3. DRAW

The DRAW architecture [8] uses an encoder-decoder architecture to generate images. A recurrent neural network is used for both the encoder and decorder. The intuition behind DRAW is that there are multiple steps in creating an image which it aims to learn in sequence. Essentially the architecture takes glimpses which are based on a attention mechanism. Unlike earlier attention mechanisms [2], the attention mechanism used in DRAW is differentiable.

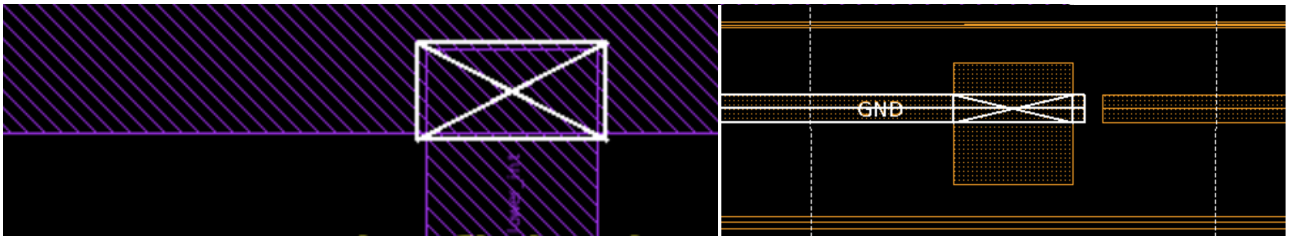
Instead of using the DRAW architecture to generate images, the encoder portion could be used to classify images based on glimpses taken.

4. Dataset and Features

There are multiple types of DRCs. We only considered DRCs which are related to congestion and cell denisty maps. These fall into two categories which are shown in Figure 6.

1. **Shorts** - When two metal layers that shouldn't overlap, end up overlapping
2. **Spacing** - When two metal layers are closer than the minimum distance that is allowed between them

Approximately 15k Congestion and Cell Density Maps were collected which were distributed among the different stages of the Physical Design flow. The images of each stage was randomly split by 0.8, 0.1 and 0.1 into training, validation and test sets.



(a) DRC violation caused by short of metal layers

(b) DRC violation caused by spacing violation

Figure 5: Types of DRC violations in consideration

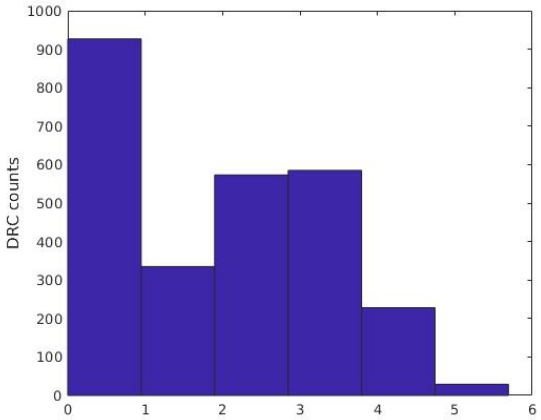


Figure 6: Histogram of log10 DRC violation numbers

4.1. Image Preprocessing

Since Congestion and Cell Density maps came on different sizes, we used the largest width and length of the dataset and padded zeros in all 3 channels to create images of the same sizes and then downsampled the images at which point the size of an image was 800x600x3. Therefore these images were relatively large than any images seen in Imagenet or CIFAR-10 datasets. Examples of the images can be seen in Figure 3.

4.2. Label Generation

The labels for each image was generated by taking the integer value of logarithm to the 10th base of each DRC violation number for each run. The population for each bin is shown in Figure 6. A large number of runs result in less than 10 violations which are attributed to the runs with good parameter setting.

4.3. Binary Classification

In order to classify a run as a fail or success, we used a threshold similar to [6] to determine if a design was manually fixable or not and hence if that specific run was successful. The threshold was set to 100 which also made the the successful vs failed run populations 50.99% to 49.01%.

5. Experimental Results

We implemented the architectures described above using Tensorflow [1]. All training was done using an internal DGX cluster with 64GBs of memory. Owing to the fact that the images were extremely large, the minibacth size was set 64. Since the amount of images were relatively small compared to ther datasets like Imagenet, we had to use k-fold cross validation with k=6.

For VGG, we used upto 6 blocks described in the methods section with convolutional layers pairs of 32,64,128,128,256 and 256 filters. Using more than 3 residual blocks did not improve the accuracy, hence we could A stride and padding of 1 was used.For DRAW, we used a glimpse size of 64x64 while taking 16 glimpses.

All training was done for 30 epochs. The starting learning rate for the VGG, resnet and DRAW networks were 0.01, 0.02 and 0.005 respectively while learning rate decay was also used.

The accuracy of DRC bin classification is shown in Figure 7. DRAW performs the best while resnet outperforms VGG om all stages of the Physical Design flow.

For VGG and resnet, cell denisty and congestion maps were trained using identical networks described above and the output fully connected layers were added together. For DRAW, only congestion maps were used.

Adding cell density maps to DRAW doesn't increase the accuracy as shown in Figure 8. This is probably because the information is redundant between the two types of images. Inspecting the Activation maps for VGG network for congestion maps show that majority of activations are due to highly congested areas (bright yellow).

We converted the DRC classification problem to a binary classification problem of a success or fail of a given run owing to a design exceeding the number of DRCs.The threshold was set to 100, which split the dataset into approximately equal fails and successes. The accuracy of binary classification increases from 6 bin DRC classification. This is primarily due to the architectures being able to correlate

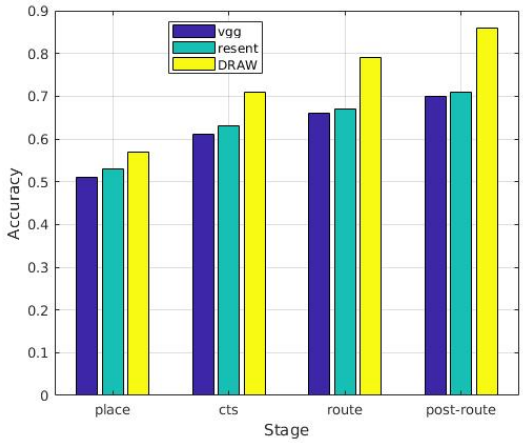


Figure 7: Accuracy for DRC bin classification

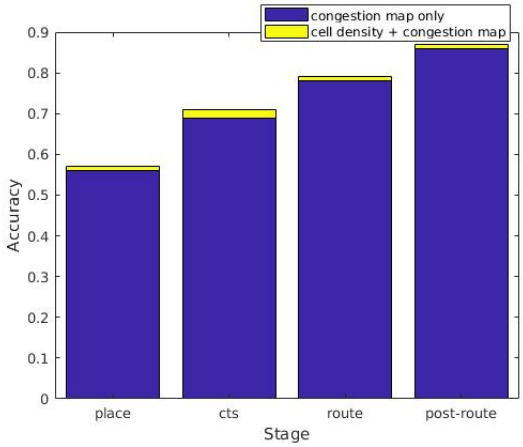


Figure 8: Increase in accuracy by adding cell density maps for the DRAW architecture

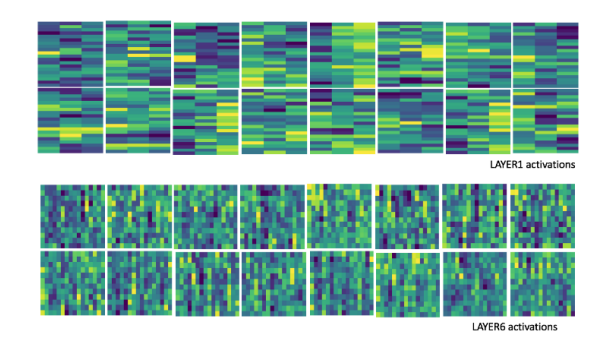


Figure 9: Activation Map for the VGG network for congestion maps

highly congested areas with higher DRC counts which can also be deduced from activation maps in Figure 9.

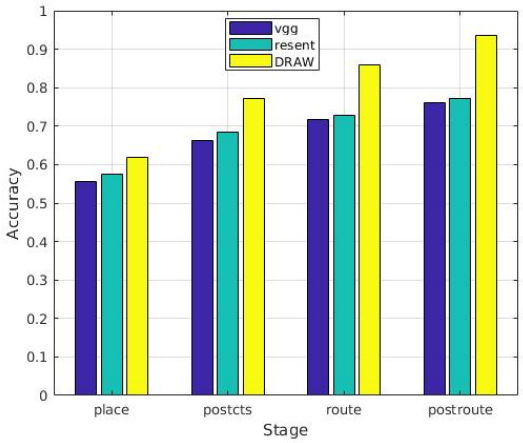


Figure 10: Accuracy for Binary classification with a threshold of 100 DRCs

Table 1: F1 scores by architecture and stage binary classification

Architecture/Stage	VGG	resnet	DRAW
placement	0.5042	0.5240	0.5635
postcts	0.6031	0.6229	0.7020
route	0.6525	0.6624	0.7811
postroute	0.6921	0.7020	0.8503

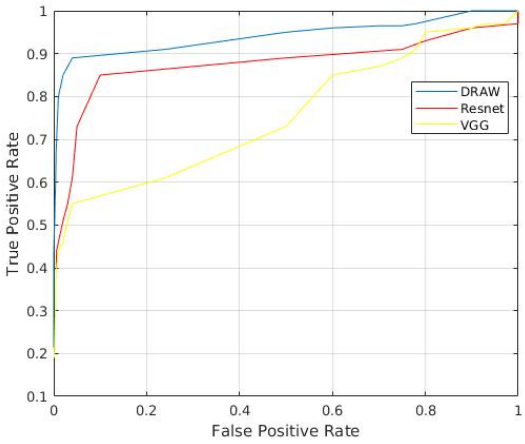


Figure 11: Receiver Operating Characterestic Curves for the architectures for binary classification

The F1 scores for binary classification is shown in Table 1. The F1 scores are slightly below the accuracy scores, but not significantly which signifies that it's extremely good at predicting failures or high DRC areas using congestion maps.

The area under the curves for the ROC curves for VGG,

resnet and DRAW are 0.741, 0.887 and 0.946 respectively which again shows that DRAW is able to classify fail/successes more effectively than the other two networks.

DRAW outperforming VGG and resnet can be explained as follows : the RNN and encoder is able to take glimpses into high congested areas and ignore areas occupied by RAMs and use that to predict the failures and DRC counts.

6. Conclusion and Future Work

Using popular deep neural net architectures, we were able to both accurately classify the number of DRCs and number if a specific run will fail.

There are certain methods, architectures that intuitively should work, but weren't used in this project due to time constraints. Therefore, we plan to do implement these in the future.

1. The cross entropy loss in bin classification should penalize certain classes/bin more than other.s If the amount of DRCs 105, the label would be that of the bin of [101-1000]. Therefore classifying it in the [11-100] should be penalized less than any other bin. We plan to modify the loss function to take into account these conditions

2. Implement other popular image classfication architectures such as densenets

3. Add additional non-image based features such as ones described [6] to improve the accuracy

4. Find a feature which can be used to normalize the DRC counts. We initially thought that the number of DRCs would be somewhat related to the size of the design (in terms of standard cell count). However as seen Figure 12, there is no direct relationship between the two features.

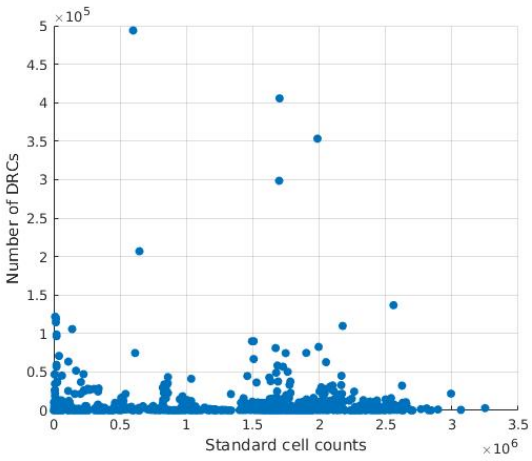


Figure 12: Relationship betweel Number of DRC violations and std cell count

7. Acknowledgements

We would like the acknowledge the use of Assignment 2 starter code in bothe CS224N and CS231N classes. We would also like to acknowledge the use of Ilya Kostrikov's DRAW implementation (https://github.com/ikostrikov/TensorFlow-VAE-GAN-DRAW)

We would also like to thank our project mentor Shyamal Buch for his valuable insight on the directions of this project.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*, 2016.

[2] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[3] A. Berg, J. Deng, and L. Fei-Fei. Large scale visual recognition challenge 2010, 2010.

[4] U. Brenner and A. Rohe. An effective congestion-driven placement framework. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(4):387–394, 2003.

[5] A. E. Caldwell, A. B. Kahng, S. Mantik, I. L. Markov, and A. Zelikovsky. On wirelength estimations for row-based placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(9):1265–1278, 1999.

[6] W.-T. J. Chan, Y. Du, A. B. Kahng, S. Nath, and K. Samadi. Beol stack-aware routability prediction from placement using data mining techniques. In *Computer Design (ICCD), 2016 IEEE 34th International Conference on*, pages 41–48. IEEE, 2016.

[7] W.-T. J. Chan, P.-H. Ho, A. B. Kahng, and P. Saxena. Routability optimization for industrial designs at sub-14nm process nodes using machine learning. In *ISPD*, pages 15–21, 2017.

[8] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[9] S. Gross and M. Wilber. Training and investigating residual nets. *Facebook AI Research, CA.[Online]. Avilable: http://torch. ch/blog/2016/02/04/resnets. html*, 2016.

[10] M. R. Guthaus, G. Wilke, and R. Reis. Revisiting automated physical synthesis of high-performance clock networks. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 18(2):31, 2013.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Con-*

ference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

[12] X. He, T. Huang, L. Xiao, H. Tian, G. Cui, and E. F. Young. Ripple: an effective routability-driven placer by iterative cell movement. In *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, pages 74–79. IEEE, 2011.

[13] M.-K. Hsu, S. Chou, T.-H. Lin, and Y.-W. Chang. Routability-driven analytical placement for mixed-size circuit designs. In *Proceedings of the International Conference on Computer-Aided Design*, pages 80–84. IEEE Press, 2011.

[14] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[15] Z.-W. Jiang, B.-Y. Su, and Y.-W. Chang. Routability-driven analytical placement by net overlapping removal for large-scale mixed-size designs. In *Proceedings of the 45th annual Design Automation Conference*, pages 167–172. ACM, 2008.

[16] A. Jiao and I. Daulagala. Logfile failure prediction using recurrent and quasi recurrent neural networks. *CS 224N, Winter 2016*, 2016.

[17] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu. *VLSI physical design: from graph partitioning to timing closure*. Springer Science & Business Media, 2011.

[18] W.-H. Liu, T.-K. Chien, and T.-C. Wang. A study on unroutable placement recognition. In *Proceedings of the 2014 on International symposium on physical design*, pages 19–26. ACM, 2014.

[19] W.-H. Liu, T.-K. Chien, and T.-C. Wang. Region-based and panel-based algorithms for unroutable placement recognition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(4):502–514, 2015.

[20] W.-H. Liu, Y.-L. Li, and C.-K. Koh. A fast maze-free routing congestion estimator with hybrid unilateral monotonic routing. In *Computer-Aided Design (ICCAD), 2012 IEEE/ACM International Conference on*, pages 713–719. IEEE, 2012.

[21] I. L. Markov, J. Hu, and M.-C. Kim. Progress and challenges in vlsi placement research. In *Proceedings of the International Conference on Computer-Aided Design*, pages 275–282. ACM, 2012.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] P. Spindler and F. M. Johannes. Fast and accurate routing demand estimation for efficient routability-driven placement. In *Proceedings of the conference on Design, automation and test in Europe*, pages 1226–1231. EDA Consortium, 2007.

[24] K. Tsota, C.-K. Koh, and V. Balakrishnan. Guiding global placement with wire density. In *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, pages 212–217. IEEE Press, 2008.

[25] M. Wang, X. Yang, K. Eguro, and M. Sarrafzadeh. Multicenter congestion estimation and minimization during placement. In *Proceedings of the 2000 international symposium on Physical design*, pages 147–152. ACM, 2000.

[26] B. Yu, D. Z. Pan, T. Matsunawa, and X. Zeng. Machine learning and pattern matching in physical design. In *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, pages 286–293. IEEE, 2015.