Accurate Wirelength Prediction for Placement-Aware Synthesis through Machine Learning

Daijoon Hyun, Yuepeng Fan, and Youngsoo Shin
School of Electrical Engineering, KAIST, Daejeon 34141, Korea

***Abstract*—Placement-aware synthesis, which combines logic synthesis with virtual placement and routing (P&R) to better take account of wiring, has been popular for timing closure. The wirelength after virtual placement is correlated to actual wirelength, but correlation is not strong enough for some chosen paths. An algorithm to predict the actual wirelength from placement-aware synthesis is presented. It extracts a number of parameters from a given virtual path. A handful of synthetic parameters are compiled through linear discriminant analysis (LDA), and they are submitted to a few machine learning models. The final prediction of actual wirelength is given by the weighted sum of prediction from such machine learning models, in which weight is determined by the population of neighbors in parameter space. Experiments indicate that the predicted wirelength is 93% accurate compared to actual wirelength; this can be compared to conventional virtual placement, in which wirelength is predicted with only 79% accuracy.**

I. Introduction

Placement-aware synthesis [1] is logic synthesis integrated with virtual P&R, and is now a common methodology for timing closure [2]. It has been motivated by increasing portion of wire delay (e.g. 35% in our experiments), which therefore should be accounted for even in logic synthesis stage. Virtual P&R is a simplified version of actual P&R: for instance, placement is done through a coarse placement without legalization and routing is performed through shortest connection without consideration on routing resources and layer assignment.

The wirelength estimated after virtual P&R is correlated to actual wirelength to some extent, but correlation is not strong enough if each individual path is compared. This is demonstrated in Fig. 1; the two wirelengths from example circuits differ by 21% on average of all paths (corresponding to dots), while there are a few dots with big difference, e.g. a path with actual wirelength of 792um and virtual wirelength of 252um, due to significantly different placement results.

In this paper, we address a methodology to estimate the actual wirelength from placement-aware synthesis through machine learning technique. A few paths of interest (e.g. top-k timing critical paths) are extracted after virtual P&R, which we simply call virtual paths. A number of parameter values that represent a virtual path are obtained, and they are processed through LDA to yield a handful of synthetic parameters. Synthetic parameters are submitted to a few machine learning models. The actual wirelength is estimated by the weighted sum of outputs from machine learning models, in which weight is determined by training samples that are neighbors of a virtual path in synthetic parameter space.

Key components in this method are input parameters and the choice of machine learning models. Initial parameters are identified for each virtual path and then synthetic parameters are compiled by using LDA, in which the importance of each synthetic parameter is modeled by its score. We find a set of synthetic parameters that correspond to the smallest error by removing the parameter with the lowest score one by one. For the choice of machine learning models, some models that correspond to a large error are first removed after a trial prediction. The best models for each training sample are marked, and we find the minimum set of models that cover at least one best model for each training sample; this problem is reduced to set cover problem, which can be formulated as integer linear programming (ILP).

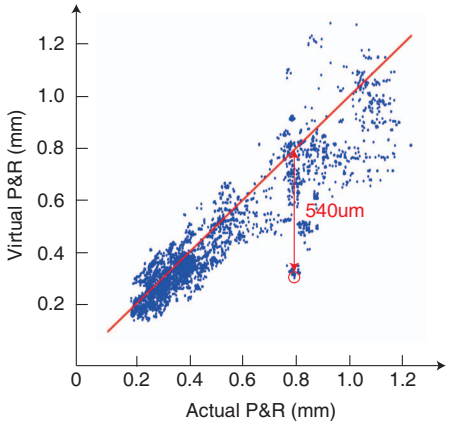

Fig. 1. Estimated wirelength after virtual P&R (y-axis) versus actual wirelength after P&R (x-axis).

II. Related Works

Some early studies have attempted to predict total wirelength and wirelength distribution in a circuit [3], [4], which can be used to estimate circuit area and approximate circuit timing, respectively. The wirelength of individual net is predicted to reflect the timing impact of wire before placement. A study uses structural parameters obtained from circuit netlist and floorplan, and actual wirelengths after P&R are fitted to polynomial function which has those parameters as variables [5]; but the error is large, about 25% on average, due to the lack of input parameters and the limitation of regression model. Linear regression with radial basis function (RBF) is also suggested [6]. But it shows only correlation coefficients in experiments; since the model uses ordinary least square without regularization, it may have large error because the model is likely to be overfitted to some outliers.

978-3-9819263-2-3/DATE19/©2019 EDAA

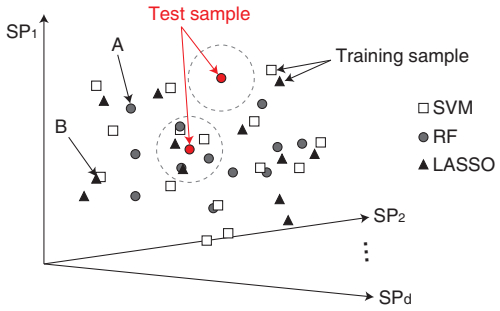

Fig. 2. Identification of nearby training samples in synthetic parameter space, in which each training sample is marked with its best models.

III. Overall Flow

In placement-aware synthesis, a test circuit is first synthesized, and its output, i.e. temporary netlist, is used for virtual P&R. We identify virtual paths that are timing critical under setup constraints after virtual P&R. The parameters that represent a virtual path are extracted, which are listed in Section IV. Each path x_i is represented by n parameters, and each parameter is normalized using z-score. We then compile synthetic parameters by $W^T x_i$, where W is $n \times d$ projection matrix obtained in Section IV, and d represents the reduced number of synthetic parameters; this matrix is calculated once with training samples, and it is applied to all virtual paths. Wirelength prediction is then performed by linearly combining the predictions of multiple machine learning models; the process of selecting models is addressed in Section V. We iteratively perform wirelength prediction for all virtual paths.

A. Wirelength Prediction Algorithm

Machine learning models, which are trained beforehand with training samples, and training samples in d-dimensional parameter space are given, where each sample is marked with the model whose accuracy is the best. A test sample goes through all the models, and their predicted values are obtained. Consider Fig. 2, where training samples are placed in synthetic parameter (SP) space, and each sample has the marking of best model; for instance, sample A in gray circle has the best accuracy in random forest (RF), while sample B has support vector machine (SVM) and Lasso regression (LASSO) as the best models. We identify the location of test sample in the space, represented with red circle, and the nearby training samples within a dotted circle are identified, in which the radius of circle is empirically determined. The predicted wirelength is obtained by using weighted sum of predictions in the models, where the weight is set to the percentage of the best model in the circle. If the circle is empty, we take the average of the predictions of all models.

IV. Parameters to Represent a Virtual Path

We extract some parameters from the circuit that a virtual path belongs to. Larger chip may have longer wires; thus, width and height are considered in addition to chip utilization. Total wirelength and the number of nets are the factors that affect congestion, which in turn affects the extent of wire detour. For the same reason, the total number of pins and the number of k-pin nets are taken into account. In addition, clock period and worst negative slack are used as reference for the timing slack of each path in the circuit.

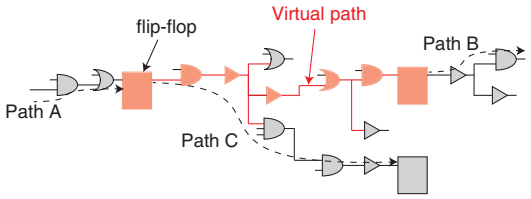

Fig. 3. An example circuit that contains a virtual path, in red color, from which input parameter are extracted.

Path-wise parameters are extracted from each virtual path. Path topology is described with some parameters, such as total wirelength, the number of cells, the total number of fanouts, and the total number of side inputs. Timing information of a path is also extracted from virtual P&R results, such as timing slack, the sum of wire delays, and wire delay ratio over path delay. The slacks of the other paths that share some cells with the virtual path are also considered. Consider Fig. 3, where a virtual path is represented with red color. Path A and B share flip-flops with virtual path, and the flip-flops can move during placement optimization if the paths are timing critical; three cells shared by path C can also be affected in the same way.

A. Synthetic Parameters

Given a matrix of training samples, $X = \{x_1, x_2, ..., x_N\}$, where x_i represents i-th sample, and N is the total number of samples. Each sample is associated with n parameters, and they are normalized using z-score. The samples are divided into multiple classes by actual wirelength in unit of 30um; it corresponds to l classes $\{c_1, c_2, ..., c_l\}$, and each class c_i has N_i samples. We now want to find a matrix W that projects X into Y with a good separability between classes.

Separability between classes is represented with a criterion function J that is the distance between projected means of classes, normalized by the within-class variability of projected samples, as shown in equation (1).

$$J(W) = \frac{|W^T S_B W|}{|W^T S_W W|}, \tag{1}$$

where variance between classes, S_B, and variance within classes, S_W, are calculated in equations (2)-(3).

$$S_B = \sum_{i=1}^{l} N_i (\mu_i - \mu)(\mu_i - \mu)^T, \tag{2}$$

$$S_W = \sum_{i=1}^{l} \sum_{x \in c_i} (x - \mu_i)(x - \mu_i)^T, \tag{3}$$

where μ_i and μ imply the mean vectors in class i and in all samples, respectively. Projection matrix W that maximizes J is obtained by finding each column vector w_i of W as shown in (4), which can be solved with Rayleigh Quotient.

$$w_i = \arg\max_{w_i} \frac{w_i^T S_B w_i}{w_i^T S_W w_i}, \quad i = 1, 2, ..., n \tag{4}$$

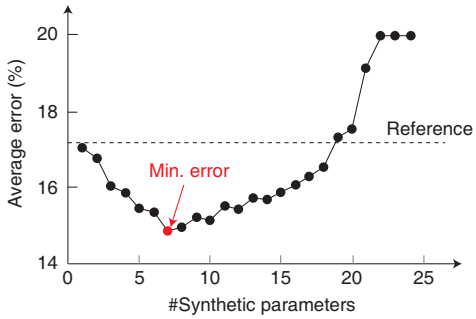

Fig. 4. The average errors for the number of synthetic parameters, in which the parameters are sorted by their score.

Resulting w_i projects each sample to i-th synthetic parameter, and its score is calculated by $J(w_i)$.

B. Selection of Minimum Number of Synthetic Parameters

Given samples are randomly divided into two groups for training and test. We build three models, SVM, ridge, and extra trees, with training group, and test group is used for prediction; the prediction errors are obtained by comparing the outputs of models to actual wirelength. One synthetic parameter with the lowest score is then removed. We repeat the training and test until there are no remaining parameters. Fig. 4 shows the average errors of wirelength prediction for the number of synthetic parameters. It has a convex shape, because a large number of parameters may cause overfitting to irrelevant parameters, and small number of parameters lead to the lack of information. We select d synthetic parameters that correspond to the minimum error ($d \leq n$), and W is reduced to $n \times d$ matrix by keeping d column vectors. In our experiment, 7 synthetic parameters are finally selected, and it results in 2.5% smaller error compared to the reference that uses 24 initial parameters before applying LDA.

V. Choice of Machine Learning Models

Wirelength prediction shown in Section III-A relies on a weighted sum of prediction from a number of machine learning models. In this section, we address which models we consider and how we choose such models.

A total of 14 machine learning models are listed here [7]–[9]. ANN is a popular model inspired by biological network; it consists of neurons and synapses, and the weights on synapses are determined in training process. SVM finds a hyperplane which minimizes the squared sum of errors in the samples out of margin; to deal with nonlinearity, RBF kernel is integrated. In addition, tree-based ensemble methods, such as random forest (RF), extra trees (ET), adaptive boosting (AdaBoost), and gradient boosting (GB), are employed. Many kinds of linear regression models are candidated. LASSO with L1 regularization and ridge with L2 regularization are considered to avoid overfitting, and Elastic net linearly combines L1 and L2 penalties. Kernel ridge uses kernel for transformation before regression. Bayesian linear regression assumes coefficients as random variables to describe uncertainty. In addition to those models, stochastic gradient descent regression, k-nearest neighbors (k-NN), and Gaussian process are considered.

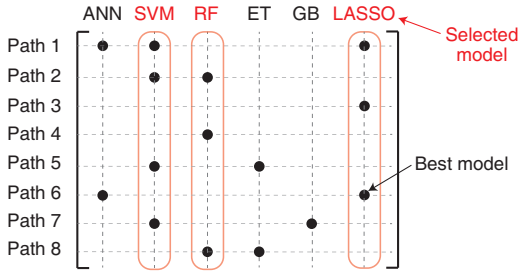

Fig. 5. Example of model selection, where at least one best model for each path is covered by three models, such as SVM, RF, and LASSO.

A. Choice of Models

Five-fold cross validation is performed with training samples, in which hyperparameters for each model are optimized; for instance, SVM with RBF kernel has two hyperparameters, regularization constant C and kernel hyperparameter γ. We identify the prediction errors for 25 combinations of them, and the best pair is selected.

As a result of validation, most models have the errors from 12% to 19%; but, k-NN and Gaussian process show relatively large errors, 33% and 53%, respectively. Two models have the common nature that the prediction is based on nearby training samples in the parameter space; thus, it causes big errors in sparse regions. Those two models are discarded from the candidate models. We then identify the best models for each sample, where the models with a difference of less than 3% from the minimum error are also included; it allows that a sample has several best models. Consider Fig. 5, where rows represent training samples that correspond to virtual paths, and columns are the candidate models. Each sample has at least one best model, marked as a black circle. In this example, if we pick three models, SVM, RF and LASSO, all samples are covered by them; this is reduced to set cover problem.

Let U be a set of N training samples. Let $s_1, s_2, ..., s_M$ be subsets of U, in which s_i is the set of samples that pick the model i as the best, and M is the number of candidate models. A set s_i is associated with a binary variable t_i, which is 1 if model i is selected, and 0 otherwise. We now formulte ILP with the objective of minimizing the sum of t_i, that is, the number of selected models. Constraint (6) ensures that every sample is present in at least one of the chosen sets.

$$\text{Minimize} \sum_{i=1}^{M} t_i \tag{5}$$

$$\sum_{i:x \in s_i} t_i \geq 1, \quad \forall x \in U \tag{6}$$

VI. Experimental Results

Experiments are carried out on a set of sequential circuits from ISCAS and ITC benchmarks as well as from OpenCores. Each circuit is compiled with synthetic 7-nm technology

TABLE I
WIRELENGTH PREDICTION ERRORS OF VIRTUAL P&R AND PROPOSED METHOD

Circuits	WL_{avg} (um)	Virtual P&R (%)	Proposed (%)
s38417	178	26	12
pci	252	23	10
mem_ctrl	262	24	12
b18	267	23	9
aes_core	268	19	7
b19	289	20	7
b14	356	26	9
b17	392	17	6
ethernet	494	17	6
rc4-prbs	815	15	4
tate	819	29	5
tate_151	1074	19	5
Average		21	7

library using commercial placement-aware synthesis [1], in which cell characteristics and wire parasitics have been appropriately scaled down from 28-nm cell library [10]. After virtual P&R, we extract the sets of parameters from top-k critical paths, where k is set to 500 and some paths with small wirelength below 100um are filtered out. As a result, 15500 samples are prepared from 31 circuits, and their actual wirelengths are obtained after actual P&R. Of these, 9500 samples of 19 circuits are used as training samples, and the remaining circuits are used for test, which are listed in the first column of Table. I.

Training samples are used for trial testing, and we select input parameters and machine learning models based on the results. Initial 24 parameters, listed in Section IV, are transformed into synthetic parameters using LDA projection, and finally 7 synthetic parameters are selected. In the model selection, 6 of 14 models, candidated in Section V, are chosen using commercial ILP solver. the selected ones are ANN, SVM, RF, LASSO, ET, and GB. All the algorithms including wirelength prediction are implemented with Python, and Scikit-Learn is used as a platform to build machine learning models.

A. Assessment of Wirelength Prediction

In Table. I, the prediction error of proposed method is shown in column 4, and it is compared to virtual P&R result in column 3. Virtual P&R has the average absolute error of 21% on average of test circuits, and the proposed method shows 7% error, which is 14% smaller than that of virtual P&R. The error of proposed method depends on the average wirelength of the circuit, listed in column 2. Machine learning techniques try to reduce the absolute distance between predictive model and the actual wirelength, not the error percentage, so the error size is not much different between short and long paths; this makes the relative error of circuits with many short paths, such as s38417, pci, and mem_ctrl, larger. For the same reason, the error for the long paths over 700um is only 3%, which is much smaller than total average error of 7%. In fact, the prediction in such long paths is more important in terms of timing impacts.

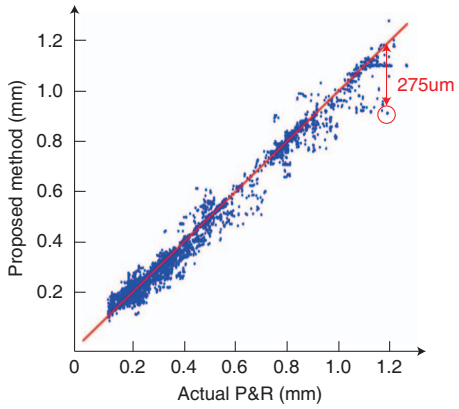

Fig. 6. Predicted wirelength with proposed method compared to wirelength after actual P&R.

Fig. 6 shows a scatter plot of test samples, which represents the actual wirelength of a path in x-axis and the corresponding predicted wirelength in y-axis. The distribution becomes much narrower and is well centered, compared to virtual P&R, illustrated in Fig. 1. The maximum error of proposed method is 275um, whose path is represented with red circle; it is much smaller than that of virtual P&R.

VII. CONCLUSION

We have addressed the method to predict the actual wirelength from placement-aware synthesis. It extracts many parameters that represent a virtual path. A reduced number of synthetic parameters are compiled through LDA projection, and they are submitted to a few machine learning models. The final prediction is given by the weighted sum of prediction from such machine learning models, in which weight is determined by the population of the best model in neighboring samples. Experiments have indicated that the predicted wirelength is 93% accurate compared to actual wirelength.

REFERENCES

[1] *Design Compiler Graphical User Guide*, Synopsys, Jun. 2016. http://solvnet.synopsys.com

[2] B. Khailany *et al.*, "A modular digital VLSI flow for high-productivity SoC design," in *Proc. Des. Autom. Conf.*, Jun. 2018, p. 72.

[3] M. Pedram, "Interconnection length estimation for optimized standard cell layouts," in *Proc. Int. Conf. on Comput.-Aided Des.*, Nov. 1989, pp. 390–393.

[4] H. T. Heineken and W. Maly, "Standard cell interconnect length prediction from structural circuit attributes," in *Proc. Custom Integr. Circuits Conf.*, Aug. 1996, pp. 167–170.

[5] S. Bodapati and F. N. Najm, "Prelayout estimation of individual wire lengths," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 6, pp. 943–958, Dec. 2001.

[6] A. Farshidi *et al.*, "A pre-placement individual net length estimation model and an application for modern circuits," *Integration, the VLSI Journal*, vol. 44, no. 2, pp. 111–122, Mar. 2011.

[7] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intelligent Syst. and Technol.*, vol. 2, no. 3, pp. 1–27, Apr. 2011.

[8] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[9] N. M. Nasrabadi, "Pattern recognition and machine learning," *Journal of Electronic Imaging*, vol. 16, no. 4, Oct. 2007.

[10] K. Han *et al.*, "ILP-based co-optimization of cut mask layout, dummy fill, and timing for sub-14nm BEOL technology," in *Proc. SPIE Advanced Lithography*, vol. 9635, Oct. 2015, pp. 1–14.