Routing-Free Crosstalk Prediction

Rongjian Liang
liangrj14@tamu.edu
Texas A&M University
College Station, TX, US

Zhiyao Xie
zx52@duke.edu
Duke University
Durham, NC, US

Jinwook Jung
jinwookjung@ibm.com
IBM T.J. Watson Research Center
Yorktown Heights, NY, US

Vishnavi Chauha
vchauhan@tamu.edu
Texas A&M University
College Station, TX, US

Yiran Chen
yiran.chen@duke.edu
Duke University
Durham, NC, US

Jiang Hu
jianghu@tamu.edu
Texas A&M University
College Station, TX, US

Hua Xiang
huaxiang@us.ibm.com
IBM Research
San Jose, CA, US

Gi-Joon Nam
gnam@us.ibm.com
IBM T.J. Watson Research Center
Yorktown Heights, NY, US

ABSTRACT

Interconnect spacing is getting increasingly smaller in advanced technology nodes, which adversely increases the capacitive coupling of adjacent interconnect wires. It makes crosstalk a significant contributor to signal integrity and timing, and it is now imperative to prevent crosstalk-induced noise and delay issues in the earlier stages of VLSI designfl ow. Nonetheless, since the crosstalk effect depends primarily on the switching of neighboring nets, accurate crosstalk evaluation is only viable at the late stages of designfl ow with routing information available, e.g., after detailed routing. There have also been previous efforts in early-stage crosstalk prediction, but they mostly rely on time-expensive trial routing. In this work, we propose a machine learning-based *routing-free* crosstalk prediction framework. Given a placement, we identify routing and net topology-related features, along with electrical and logical features, which affect crosstalk-induced noise and delay. We then employ machine learning techniques to train the crosstalk prediction models, which can be used to identify crosstalk-critical nets in placement stages. Experimental results demonstrate that the proposed method can instantly classify more than 70% of crosstalk-critical nets after placement with a false-positive rate of less than 2%.

CCS CONCEPTS

• **Hardware** → **Very large scale integration design**; • **Computing methodologies** → *Machine learning approaches.*

KEYWORDS

crosstalk, routing-free, machine learning

ICCAD '20, November 2–5, 2020, Virtual Event, USA

ACM ISBN 978-1-6654-2324-3/20/11…$15.00
https://doi.org/10.1145/3400302.3415712

ACM Reference Format:
Rongjian Liang, Zhiyao Xie, Jinwook Jung, Vishnavi Chauha, Yiran Chen, Jiang Hu, Hua Xiang, and Gi-Joon Nam. 2020. Routing-Free Crosstalk Prediction. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '20), November 2–5, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3400302.3415712

1 INTRODUCTION

Crosstalk-induced noise and delay variation have emerged as a serious concern in advanced technology nodes. Through capacitive coupling, a signal switching of one net causes crosstalk noise at its neighboring nets [27]. Noise amplitude can even reach up to 30% of V_{DD} [21], which may exceed the threshold voltage of transistors and lead to glitches, which impose a risk of logic errors and unwanted switching power consumption. Moreover, the coupling capacitance itself serves as extra load increasing both signal delay and internal power dissipation; the incremental delay due to coupling capacitance along a timing path can reach 300ps [22], comparable to clock periods of modern high-performance processors.

Nontheless, an accurate estimation of crosstalk effects is only possible after exact routing topology is available, i.e., after detailed routing. As it is among the last few steps in physical design, few rooms may remain tofi x all the crosstalk-induced design problems even if we identify every crosstalk issue [27]. Morefl exibility for design changes can be found at global routing stage [18, 30]. Estimating crosstalk, however, becomes much harder at this point as exact routing topology is not determined yet. There may also be still insufficient opportunities tofi x significant crosstalk-induced problems even at the global routing stage.

In this regard, many research efforts have been undertaken to estimate and mitigate crosstalk problems at earlier design stages, e.g., placement [16, 22]. There is usually a greater degree of design flexibility at earlier stages, which helps us resolve all the significant crosstalk issues if identified. The key problem of such an early-stage estimation, however, is the absence of routing information. It is mostly impossible to estimate crosstalk accurately with placement alone. Indeed, crosstalk driven placement [16, 22] resorts to global routing or trial routing for obtaining an approximate estimate of

routing landscape and thereby capacitive coupling of nets. An evident drawback here is that global or trial routing is time-consuming and thus induces huge runtime costs.

1.1 Motivation

There is a tradeoff between accuracy of crosstalk prediction and design flexibility across three main relevant stages: placement, global routing, and detailed routing. An ideal crosstalk avoidance flow could be such as:

(1) During placement, significant crosstalk risks are identified and resolved by leveraging the greater design flexibility.
(2) Most of the remaining crosstalk risks are eliminated in global routing via layer assignment [28] and area routing [25].
(3) In detailed routing, complete crosstalk avoidance is achieved with the precise crosstsalk evaluation.

While crosstalk avoidance at routing stages has been well studied [18, 25, 27, 30], the placement stage solutions [16, 22] are still far from being practical largely due to the dependence on trial/global routing, which can easily take more than a half hour for a modern design. Indeed, pre-routing crosstalk prediction is a major weak link in realizing the ideal crosstalk avoidance flow.

One might consider employing existing routing congestion prediction methods [29] and use the results as a proxy of predicted crosstalk hotspots, since coupling capacitance correlates with routing congestion. Despite the correlation, Figure 1 shows there exists difference between the routing congestion hotspot and the crosstalk-induced noise hotspot. One important reason for the mismatch is that noise is determined by not only coupling capacitance, but also electrical and logical parameters.

1.2 Contributions

In this paper, we present a predictive method that can quickly identify a majority of crosstalk prone nets at placement stages, *without any routing information*. Such predictive identification would not only assist crosstalk driven placement [16, 22], but also make other crosstalk mitigation techniques viable at placement stage, such as gate sizing [4] and buffer insertion [3]. Note that it is not necessary to identify all problematic nets as the goal of placement-stage techniques is to reduce crosstalk risk and make subsequent avoidance techniques feasible rather than completely solve all crosstalk problems. At the same time, the prediction must be very fast, at least an order of magnitude faster than global/trial routing.

We identify routing and net topology-related features, together-with electrical and logical features, which affect crosstalk-induced noise and delay. Our approach leverages recent progresses on machine learning (ML) such as XGBoost [7]. In addition, graph-based ML techniques including GraphSage [12] and graph attention networks [26] are investigated for the crosstalk prediction. Given a cell placement solution for a design, the proposed approach predicts coupling capacitance, peak noise and crosstalk induced incremental delay for every signal net. Experimental results show that it can identify over 70% of nets with top crosstalk problems at false positive rate no greater than 2%. At the same time, its computation speed is two orders of magnitude faster than a conventional approach based on global routing. Detailed analysis of feature importance is also performed. To the best of our knowledge, this is the first approach to routing-free crosstalk prediction.

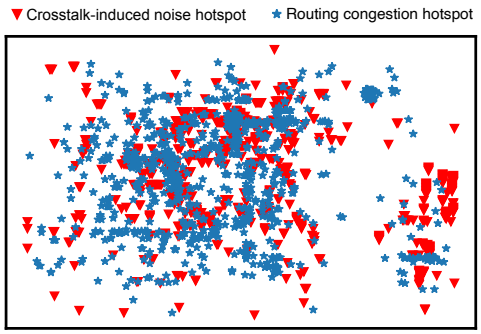

Figure 1: Routing congestion and crosstalk-induced noise hotspots.

2 RELATED WORK

Due to the importance of crosstalk avoidance, crosstalk estimation has been extensively studied in the past [1, 6, 15, 21]. Regardless accuracy and computation speed, all the previous methods require wire adjacency information, which is not available until detailed routing [27] or area routing [25]. Crosstalk estimation has also been studied targeting global routing [5, 18] and layer assignment stages [28]. Since no wire adjacency is available at these stages, the number of wire segments in a global routing cell is instead used to infer crosstalk in a probabilistic manner.

Crosstalk avoidance has also been addressed at placement [16, 22]. Such methods employ trial or global routing to get rough routing topology from placement. Based on the routing congestion information from the rough routing, coupling capacitance is estimated through curve-fitting from data of previously completed designs. Based on the estimated coupling capacitance and the routing routing topology, crosstalk noise is then computed using a simplified model, e.g., [22]. There have also been attemps to address crosstalk in even earlier design stages including technology mapping [10] and high-level synthesis [23]. Crosstalk avoidance is particularly investigated for bus design [9] where wire permutation is decided.

An ML-based crosstalk estimation is introduced in [13], to take crosstalk induced incremental delay into account within STA of routed designs. Instead of relying on the time-consuming signal integrity (SI) modes of STA tools, an ML model accounts for the effect of crosstalk during STA without turning on the SI mode. However, it can be only applied to routed designs. To the best of our knowledge, there is no previous work that can predict crosstalk without using any routing information. This is a void to be filled by our work for addressing crosstalk at placement and post-placement optimizations, such as gate sizing and buffer insertion.

3 METHODOLOGY

3.1 Overall flow

Crosstalk prediction by machine learning (ML) classification is a more attainable approach than ML regression. Moreover, classification results usually suffice for crosstalk avoidance in early

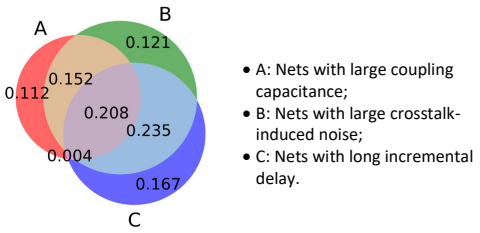

Figure 2: The Venn diagram of three crosstalk-critical net categories. The value on each region shows the proportion of the nets belonging to the region, normalized against the total number of crosstalk-critical nets.

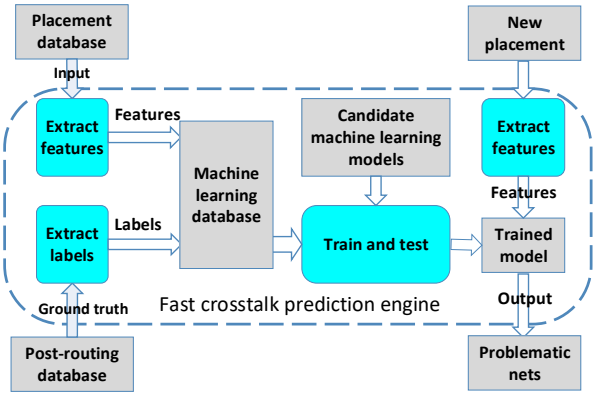

Figure 3: Crosstalk modeling flow.

design stages. We target at solving three classification tasks at the placement stage, to identify

(1) The nets likely to have large coupling capacitance.
(2) The nets likely to have large crosstalk-induced noise.
(3) The nets likely to have long incremental delay due to crosstalk.

Figure 2 shows the Venn diagram of the above three sets of crosstalk-critical nets according to our experiment data. We can find that these sets have overlaps but they are not identical.

The crosstalk modeling flow is shown in Figure 3. Input features and ground truth information are extracted from the placement and the post-routing databases, respectively, and they further constitute the ML database. By training and evaluating prediction performance of candidate ML models, e.g., XGboost, with the labeled data stored in the ML database, the most effective feature sets and the best models for three crosstalk classification problems are determined, which can be used to fast identification of problematic nets in new placement instances.

The raw input to crosstalk prediction engine includes placed DEF file, standard cell libraries, and the static timing analysis (STA) results generated after placement. The placed DEF contains the circuit netlist, and the locations of cells and input/output ports after placement. The standard cell library files are used to get the physical, electrical and logical properties of the cells in the circuit. The STA results generated after placement give timing information such as cell delays, wire delays, and transition times. Net-based features are extracted from these files and then fed into ML models.

The ground truth information is extracted from the parasitic information file and the timing report in SI mode generated after detailed routing. For coupling capacitance prediction, the ground truth of a placement sample is represented by a binary vector of length n, where n is the total number of nets in a design. Each entry in the vector corresponds to a net, indicating whether its total coupling capacitance is larger than a given threshold. Crosstalk noise is measured at receiver pins. For each net, we calculate the peak noise amplitude and noise width product, denoted by AW, at the sink pins, because the noise susceptibility of digital logic gates is usually characterized by a noise amplitude pulse width plot. The ground truth of a placement for noise prediction is denoted by a binary vector of length n. Similarly, we obtain a vector denoting whether the nets in a placement have crosstalk-induced incremental delays larger than a given threshold.

3.2 Feature Selection

3.2.1 **Probabilistic Congestion Estimation.** Routing congestion strongly correlates with crosstalk, since coupling capacitance tends to occur in congested areas. A probabilistic technique for congestion analysis [24] is employed in our work due to its great runtime advantage over other techniques and its good correlation with the post-routing solution.

Rectangular Uniform wire DensitY (RUDY) of a net [24] is obtained via dividing the total wire volume going through the bounding box of the net by the bounding box area. We call two nets neighbors if their bounding boxes overlap. For the i-th net in a placement, with its neighboring nets denoted by $N(i)$, the total wire volume $\text{wireVolume}(i)$ that go through its net-bounding-box is calculated as

$$\text{wireVolume}(i) = \text{HPWL}(i) + \sum_{j \in N(i)} \text{HPWL}(j) \frac{\text{overlapArea}(i,j)}{\text{area}(j)}, \quad (1)$$

where $\text{HPWL}(j)$ is the half perimeter wire length (HPWL) of net j, $\text{area}(j)$ is the bounding box area of net j, and $\text{overlapArea}(i,j)$ is the overlap area of the bounding boxes for nets i and j. The RUDY of the i-th net is then given by

$$\text{RUDY}(i) = \frac{\text{wireVolume}(i)}{\text{area}(i)}. \quad (2)$$

Nets can be divided into long-range nets and short-range nets according to their HPWL[1]. In [29], it is shown that routing congestion has a stronger correlation with long-range nets than with shorter ones. In this regard, we also extract a feature, which we call **longRangeRUDY**, in a similar way as (1) and (2). The difference is that only long-range nets are taken into account when computing the total wire volume.

3.2.2 **Net Topology Estimation.** Net topology is necessary for crosstalk estimation. Because different routing topology of a net exposes it to different aggressors. Also, it leads to different electrical characteristics of interconnects, which have impacts on crosstalk noise and incremental delay. A few net-topology-related features are extracted as follows.

[1]We used 25μm for the long-range net threshold in this work.

- **HPWL** of the net-bounding-box.
- **area** of the net-bounding-box.
- **fan-out** of the source, i.e., the number of sinks of the net.
- **max-ss-distance**: the maximal distance between the source cell and the sink cells.

3.2.3 **Product of the Wire-length and Congestion.** As illustrated in [16], the total coupling capacitance of a net is proportional to the product of its wire-length and the unit coupling capacitance. We use the product of the HPWL and the RUDY/long-range-RUDY of a net as an indicator of its coupling capacitance.

- **HPWL-RUDY/HPWL-longRangeRUDY**: the product of HPWL and RUDY/longRangeRUDY of a net.

3.2.4 **Electrical and Logical Features.** The electrical properties of cells play an important role in crosstalk. For example, the crosstalk noise is affected by the source cell's resistance [22]. Also, the logic type of the source cell may affect the switching activity of the net, and consequently affect the noise and incremental delay. However, given the complicated timing models used by modern cell libraries, it is difficult to capture all crosstalk-related properties of a cell. To address this problem, we propose to use a logic-based encoding, along with the output capacitance, to represent a cell. First, library cells are consolidated into groups according to logic types. Each group contains cells with the same logic, but can have different fan-in count (e.g., a two input NAND and a three input NAND belong to the same group) and different sizes. We use a variant of the one-hot encoding to encode the gate groups. Specifically, a vector of length N_g, the total number of gate groups, with only one non-zero entry is used to describe which group the cell belongs to. Unlike assigning 1 to the non-zero entry in the one-hot encoding, we assign the cell's fan-in to the non-zero entry. An additional feature, output capacitance, is used to capture the size of the gate. A few other electrical and logical structure features are also extracted.

- l_0 to l_{N_g} : the logic-based encoding of the source cell.
- **sourceCap**: output capacitance of the source cell.
- **sinkCap**: the sum of the input capacitance of sink cells.
- **fan-in**: the fan-in of the source cell.

3.2.5 **Timing Information.** The pre-routing timing report gives a rough estimation of wire delays and slews, which is informative for crosstalk prediction. For example, smaller slew often means stronger drive strength, consequently more resistant to aggregator's effect in term of incremental delay.

- **wireDelay** : the longest wire delay from the source to the sinks. Note it is a rough prediction from the pre-routing STA.
- **outputSlew** of the source cell from the pre-routing STA.

We also tried clock period, toggle rate and other timing related features. But we have not observed improvement brought by these features in our experiments.

3.2.6 **Neighboring Net Information.** Crosstalk noise and incremental delay depend not only on coupling capacitance, but also on the coupling location [22] (near the source or sink cells) and the aggressors' drive strength. We estimate the coupling location via the relative location of the net's bounding box and its neighbors' bounding boxes.

- **#Neighboring nets**: number of neighboring nets.
- **#Neighboring long-range-nets**: number of neighboring long-range-nets.
- **mean-, std-, max-overlapArea**: the average/standard deviation/maximum of overlap areas between the net's bounding-box and neighboring nets' bounding-boxes.
- **mean-dist-source-overlap**: the average distance between the source cell and the geometric centers of overlap regions.
- **weighted-dist**: the average distance between the source cell and the geometric centers of overlap regions, weighted by the area of each overlap region.
- **dist-source-maxOverlap**: the distance between the source cell and the geometric center of the largest overlap region.

We use the sourceCap and outputSlew to represent the drive strength of a net. If a net is surrounded by nets with strong drive strength, then its aggressors is likely to have strong drive strength. The following features are employed to capture neighboring nets' drive strength.

- **mean-, std-sourceCap**: the average/standard deviation of sourceCap of neighboring nets.
- **sourceCap-maxOverlap**: the sourceCap of the neighboring net that has the largest overlap with the target net.
- **mean-, std-outputSlew**: the average/standard deviation of outputSlew of neighboring nets.
- **outputSlew-maxOverlap**: the outputSlew of the neighboring net that has the largest overlap with the target net.

Note that we do not use the electrical and logical features, and timing information in coupling capacitance prediction, since coupling capacitance is mainly determined by layout information.

3.3 Machine Learning-Based Crosstalk Estimation Models

Four popular ML techniques, logistic regression, neural network, random forest and XGboost, were used to model the mappings from the extracted features to coupling capacitance, crosstalk-induced noise and incremental delay. For each technique, three independent models were trained and fine-tuned for the three classification tasks. Implementation details about the models are shown as follows.

- **Logistic regression (LR)**: It is a linear classification model. We used scikit-learn [20] for the implementation, employing L_2 regularization to mitigate overfitting.
- **Neural network (NN)**: Multilayer perceptron neural-networks, implemented via the Pytorch [19], were employed. We implemented a narrow network with 3 layers for coupling capacitance prediction, and used deeper networks with 5 layers for crosstalk noise prediction and incremental delay prediction, since the mappings from input features to crosstalk noise and to incremental delay are considered to be more complex than the mapping from input to coupling capacitance. We also fine-tuned learning rate and other hyper-parameters to achieve the best performance.
- **Random forest (RF)**: A random forest consists of independently-trained decision trees, and the classification results are obtained by averaging the decisions of all trees or by voting [14]. Larger number of trees and deeper trees can make the model more expressive, but may result in overfitting; if a forest is too small and narrow, it may fall into underfitting. We tuned these parameters

carefully to achieve balance between overfitting and underfitting. In our experiment, there were 100 trees with the maximum depth of 5 in the random forest implementation.

- **XGboost (XG)**: We employed the Gradient Boosted Regression Trees (GBRT) available in XGboost [7]. In GBRT, model trees are trained sequentially, where each training depends on the errors of previous ones. The final decision is the sum of decisions of all trees. In our experiment, the best results were obtained by 25 trees with the maximum depth of 8.

We also investigated the effectiveness of graph-based ML techniques for the crosstalk prediction. Two net-centric graphs were generated: A dual circuit graph (DCG) and a proximity graph (PG). In a DCG, each node v_i represents a net, and a directed edge exists between a pair of nets (v_i, v_j) if a sink of v_i is the source of v_j. It can capture electrical connection information. A PG is used to capture the physical proximity among nets in the layout. Each node represents a net, and a unidirectional edge is created between two nets if the overlap area of their bounding boxes are larger than a given threshold[2]. The weight of edge is the area of the overlap region. Based on these two graphs, three ML models were implemented.

- **GraphSAGE with NN (on DCG)**: GraphSAGE [12] is a popular convolutional operation on graphs, which can learn a representation for each node integrating the information of the node itself and from its neighborhood. We used GraphSAGE operations on a DCG to integrate the information among electrically-connected nets, which is useful for crosstalk prediction. In particular, two GraphSAGE layers, implemented by Pytorch Geometric [11], were used to propagate electrical and logical features on the DCG, and output a learned feature vector for each net. Then the feature vector was concatenated with other features proposed in Section 3.2 and fed to a NN for final classification.
- **GraphSAGE with XG (on DCG)**: It is the same as the above, except that the final classification is done with an XGBoost model instead of a NN.
- **NN with GraphAttention (on PG)**: GraphAttention [26] is an operation on graphs, suitable to capture dependency of neighboring nodes. For crosstalk prediction, there may be dependency among neighboring nets, since crosstalk occurs between physically-close nets. We use GraphAttention operations to capture such dependency on a PG. Firstly a multilayer perceptron neural network was applied to each net to output a rough classification result. Then two GraphAttention layers, implemented using Pytorch Geometric, were used to propagate rough classification result on the PG to get the final classification result.

4 DESIGN OF EXPERIMENTS

Our experiments were conducted on 12 designs from the IWLS 2005 benchmarks [2]. We generated a total of 304 placements with over 3 million labelled nets, by running a commercial design flow with various yet realistic parameter settings. Logic synthesis was performed using Synopsys Design Compiler and physical design was conduced with Cadence Innovus. We used ASAP7 standard cell library [8], an open-source 7-nm library. The clock period varied from 250ps to 400ps. The average number of nets and registers after placement as well as runtime of global routing (with the globalRoute command in Innovus) for each design are listed in Table 1. After detailed routing and RC extraction, STA was performed by Synopsys PrimeTime-SI and the analysis results were used as ground truth. The synthesis flow was conducted on a Linux server equipped with an Intel Xeon E5-2680 CPU.

[2]We used 15μm^2 for the threshold in this work.

Table 1: Benchmarks used in our experiment. The groups are for training/testing data partitioning (Section 4.1.1).

Design	#Nets	#Registers	GR Runtime	Group ID
systemcdes	3590	190	3s	1
tv80	4950	353	9s	2
systemcaes	5308	685	13s	3
mem_ctrl	6473	936	8s	4
wb_dma	6936	486	5s	4
ac97_ctrl	8397	1820	9s	3
usbf_funct	8629	132	13s	2
fpu	14416	542	41s	1
pci	19318	174	20s	1
aes_core	37105	527	21s	2
ethernet	47051	10002	109s	3
vga_lcd	74467	16038	203s	4

The ground truth of coupling capacitance, crosstalk-induced noise and incremental delay are real numbers. They were transformed into binary classification labels via pre-defined thresholds (called *ground truth thresholds*). The thresholds used for coupling capacitance, noise and incremental delay classifications were 0.5fF, $2V$ps and 2ps, respectively[3].

4.1 Model Training and Testing

4.1.1 Training & Testing Schemes. For the labelled data that we can obtain, two different partition schemes between training and testing sets were applied for the model evaluation.

- **Scheme 1:** Testing on nets from unseen placement instances. For each design, we randomly chose three placement solutions and used their labelled net samples as testing data, while nets from other solutions were used as training data. Please note testing placement instances may involve some same designs as those in the training set, although their placements were different.
- **Scheme 2:** Testing on nets from unseen designs. The 12 designs were divided into four groups according to their number of nets, as shown in Table 1, making the four groups have similar amount of net samples. Four-fold cross-validation was performed. To be specific, four rounds of experiments were ran and their results were averaged. In each round, all net samples from three groups were taken as training data, while nets from the only remaining group were used as testing data. Each round had distinct testing designs. This is a stricter testing scheme than scheme 1 as not only the testing placement instances but also the testing designs are completely unseen during training.

4.1.2 Baseline Method. A global routing-based method similar to [22] was implemented as the baseline method. We ran global routing

[3]Coupling capacitance of 0.5fF is about 60% of the input capacitance of a buffer in the ASAP7 library. Also, 2ps is about 1% of the clock period. In Section 5.5, we investigate how classification performance changes with the ground truth thresholds.

to get congestion of each global routing cell (gcell). Then, we generated coupling capacitance of each gcell $C(i, j)$ via curve-fitting as in [22]. The total coupling capacitance of each net CC was estimated to be $CC = \sum_{i,j} C(i, j)$, where (i, j) is a gcell traversed by the net. The noise of each net N was estimated as $N = \sum_{i,j} R_1(i, j)C(i, j)$, where $R_1(i, j) = a_1 d(i, j) + b_1$ and $d(i, j)$ is the Manhattan distance from the driver to (i, j). Similarly, the incremental delay was estimated as $D = \sum_{i,j} R_2(i, j)C(i, j)$, where $R_2(i, j) = a_2 d(i, j) + b_2$. The coefficients a_1, b_1, a_2 and b_2 were obtained via curve-fitting. The CC, N and D were then used to identify problematic nets.

4.2 Performance Metrics

The proportion of crosstalk-critical nets to the total nets in a design is typically small. The positive samples (i.e., crosstalk-critical nets) in our classification problems is hence much less than the negative samples in general. In this regard, we evaluate prediction performance with the following four metrics widely used for imbalanced data set [17], based on TP (the number of true positive samples), FP (false positive), TN (true negative) and FN (false negative).

- False positive rate: FPR = $\frac{FP}{FP+TN}$.
- True positive rate: TPR = $\frac{TP}{TP+FN}$.
- F1 score: F1 = $2 \times \frac{TPR \times Precision}{TPR+Precision}$, where Precision = $\frac{TP}{FP+TP}$.
- Balanced accuracy: BAcc = $\frac{(1-FPR)+TPR}{2}$.

The raw output of our crosstalk prediction engine is a scalar in $[0, 1]$ for each net, which can be viewed as the probability of the net being crosstalk-critical in terms of coupling capacitance, noise or incremental delay. A classification threshold is required to turn the raw output into binary prediction label. A high threshold will lead to high TPR, but also high FPR; otherwise the vice. Receiver Operating Characteristic (ROC) curve indicates the trade-off between TPR and FPR varying the classification thresholds. A large area under curve of ROC-curve (AUC of ROC), implies better prediction performance. It is 1 for the perfect prediction and 0.5 for a random guess.

5 RESULTS

5.1 Crosstalk Prediction: Scheme 1

The classification results from Scheme 1 are shown in Tables 2, 3, and 4. Overall, XGboost models achieved the best performance, even beating the baseline method in all the three classification tasks, in terms of most of the performance metrics we used. The random forest models achieved the second best results.

For coupling capacitance prediction, the four modeling techniques gave similar results. It means that there is a strong correlation between the extracted features and the coupling capacitance of nets. Thus, even the linear classification model, logistic regression, can capture such correlation very well. However, the performance of logistic regression was obviously inferior to non-linear techniques in the noise and incremental delay prediction, especially in the latter. It suggests the mappings from extracted features to crosstalk-induced noise and incremental delay are largely non-linear, which can not be captured well by linear models. One other observation is that ensemble tree-based modeling, i.e., random forest and XGboost, outperformed the neural network approach in our tasks.

Table 2: Coupling capacitance prediction results (Scheme 1).

Model	FPR	TPR	F1	BAcc	AUC
Baseline	1.10%	79.49%	52.67%	89.20%	0.993
LR	1.04%	81.98%	47.64%	90.47%	0.905
NN	1.01%	81.65%	48.24%	90.32%	0.994
RF	1.01%	82.45%	48.48%	90.72%	0.993
XG	0.94%	83.17%	50.58%	91.12%	0.994

Table 3: Crosstalk noise prediction results (Scheme 1).

Model	FPR	TPR	F1	BAcc	AUC
Baseline	1.30%	50.45%	40.19%	74.58%	0.974
LR	1.38%	68.99%	43.40%	83.81%	0.838
NN	1.23%	67.51%	44.85%	83.14%	0.987
RF	1.16%	68.82%	46.77%	83.83%	0.987
XG	0.99%	70.25%	50.65%	84.63%	0.990

Table 4: Incremental delay prediction results (Scheme 1).

Model	FPR	TPR	F1	BAcc	AUC
Baseline	4.04%	57.18%	21.18%	76.57%	0.947
LR	2.08%	61.44%	30.70%	79.68%	0.797
NN	2.08%	70.85%	34.58%	84.38%	0.980
RF	2.03%	72.78%	35.80%	85.37%	0.978
XG	1.88%	77.01%	39.17%	87.56%	0.986

Table 5: Coupling capacitance prediction results (Scheme 2).

Model	FPR	TPR	F1	BAcc	AUC
Baseline	1.06%	78.93%	53.14%	88.94%	0.993
LR	1.03%	80.91%	50.98%	89.94%	0.899
NN	0.98%	80.73%	51.90%	89.87%	0.959
RF	0.90%	81.62%	54.16%	90.36%	0.994
XG	0.89%	81.87%	54.62%	90.49%	0.994

Table 6: Crosstalk noise prediction result (Scheme 2).

Model	FPR	TPR	F1	BAcc	AUC
Baseline	1.56%	50.88%	37.47%	74.66%	0.970
LR	1.62%	75.15%	47.17%	86.77%	0.868
NN	1.45%	66.56%	45.09%	82.55%	0.846
RF	1.31%	76.34%	52.27%	87.51%	0.987
XG	1.28%	78.37%	53.69%	88.54%	0.990

Table 7: Incremental delay prediction results (Scheme 2).

Model	FPR	TPR	F1	BAcc	AUC
Baseline	4.35%	52.83%	18.71%	74.24%	0.937
LR	2.43%	69.29%	33.19%	83.43%	0.834
NN	2.39%	73.10%	35.07%	85.35%	0.939
RF	2.10%	74.74%	38.45%	86.32%	0.978
XG	1.80%	76.25%	42.35%	87.23%	0.986

Figure 4 shows the trade-off between TPR and FPR in XGboost models. It can be seen that the coupling capacitance, crosstalk-induced noise and incremental delay predictions are progressively more difficult.

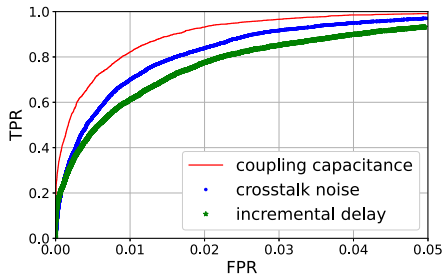

Figure 4: ROC curves of the XGboost prediction results.

5.2 Crosstalk Prediction: Scheme 2

Results from Scheme 2 are similar to those from Scheme 1. Among the four modeling techniques, XGboost achieved the best performance in all three classification tasks. Detailed results for coupling capacitance, noise and incremental delay predictionns are listed in Tables 5, 6 and 7, respectively. Note that Scheme 2 is stricter than Scheme 1 in that not only the testing placement solutions but also the testing designs are completely unseen during training. But the prediction results from Scheme 2 were similar to those from scheme 1, meaning the proposed modeling generalized well on unseen designs.

5.3 Crosstalk Prediction: Graph-Based Models

For coupling capacitance prediction, we did not adopt "GraphSAGE + NN" and "GraphSAGE + XG" models, which act on the DCG, since coupling capacitance does not depend on electrical connection. The TPR of the "NN + GraphAttention" model was approximately 1% higher than that of the NN model, at the same FPR, but still inferior to the XGboost model.

Tables 8 and 9 show the results of graph-based models for crosstalk-induced noise and incremental delay prediction using Scheme 1. It seems that graph-based learning techniques did not help improve prediction performance. The reason might be that electrical connection information is not significant for crosstalk prediction at early stages, so the DCG is not helpful; physical proximity information can be well captured by extracted features, such as RUDY, so it might not be necessary to use the PG. Among the graph-based techniques the GraphSAGE + XGboost model achieved the best result, similar to that of the XGboost method. The results of these two approaches were also fairly consistent. In particular, 94% and 95% of the classified positive samples were identical in the two models, for noise and incremental delay predictions, respectively.

5.4 Feature Importance Analysis

One important advantage of tree-based models over neural network models is that they are more interpretable. After training a tree-based model, users can check which features are most important in building the decision trees. Importance can be defined from various aspects. One commonly-used metric is "Gain", which is the improvement in accuracy brought by a feature. Figure 5 shows the top-10 important features in coupling capacitance, crosstalk-induced noise and incremental delay predictions, in term of "Gain".

Table 8: Results of graph-based models with dropping some features for noise prediction (Scheme 1).

Model	FPR	TPR	F1	BAcc	AUC
XG	0.99%	70.25%	50.65%	84.63%	0.990
GS + NN	1.10%	63.27%	44.89%	81.09%	0.987
GS + XG	1.03%	70.91%	50.22%	84.94%	0.990
NN + GA	1.18%	65.89%	44.92%	82.36%	0.988
Layout only	1.26%	57.45%	39.10%	78.09%	0.981
No timing info.	1.01%	69.40%	49.86%	84.19%	0.989
Binary encoding	0.99%	69.78%	50.34%	84.39%	0.990

Table 9: Results of graph-based models with dropping some features for incremental delay prediction (Scheme 1).

Model	FPR	TPR	F1	BAcc	AUC
XG	1.88%	77.01%	39.17%	87.56%	0.986
GS + NN	2.09%	62.70%	31.15%	80.31%	0.822
GS + XG	1.95%	76.77%	38.23%	87.41%	0.984
NN + GA	1.94%	65.27%	33.65%	81.67%	0.861
Layout only	2.66%	61.50%	26.28%	79.42%	0.966
No timing info.	2.09%	73.45%	35.45%	85.68%	0.983
Binary encoding	1.90%	76.39%	38.74%	87.25%	0.986

It can be seen that the layout features played an important role in the three crosstalk prediction because crosstalk heavily depends on layout. It is noteworthy that HPWL-longRange-RUDY was even more important than HPWL-RUDY. It seems that crosstalk is more relevant to long-range-nets than to short-range-nets. As for crosstalk-induced noise and incremental delay prediction, we can see that electrical features (e.g., sourceCap and sinkCap), logical features (e.g., logic-based encoding for the source cell: l_6, l_{10} and l_{11}), the timing information (e.g., outputSlew) and the neighboring net information (e.g., the weighted-dist and the max-dist-source-overlap) also have great importance. To better show their importance, we conducted additional experiments where only layout features were used. From the last three rows of Table 8 and Table 9, we can find that prediction accuracy dropped significantly without electrical and logical features as well as timing information. Moreover, we explored the effects of dropping timing information alone, i.e., the wire delay and slew-related features. It is interesting that dropping these features does not hurt the noise prediction, but does hurt incremental delay prediction noticeably. Also, we found a small decrease in prediction performance if we use the conventional binary encoding of cells instead of the proposed encoding.

5.5 Impact of Ground Truth Threshold and Training Sample Counts

We investigated how F1-score and AUC of ROC change with the ground truth thresholds using XGboost. In Figure 6, we can find that a higher ground truth threshold led to a larger AUC of ROC. For the prediction of crosstalk-induced noise and incremental delay, however, the 0.5× threshold resulted in the highest F1-score.

If we train one classification model with the original threshold and train another one with the 2× threshold, their prediction results might not be consistent, in the sense that a net might be predicted larger than the 2× threshold by one model but predicted smaller

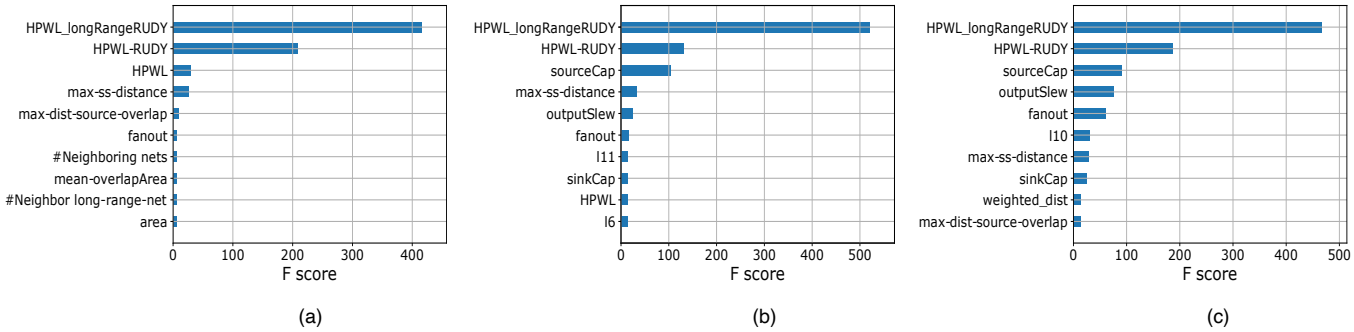

Figure 5: Top-10 important features: (a) in coupling capacitance, (b) in crosstalk noise, and (c) in incremental delay predictions.

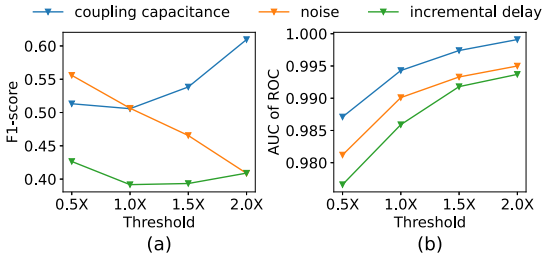

Figure 6: Impact of ground truth threshold in XGboost. (a) F1-score, and (b) AUC of ROC.

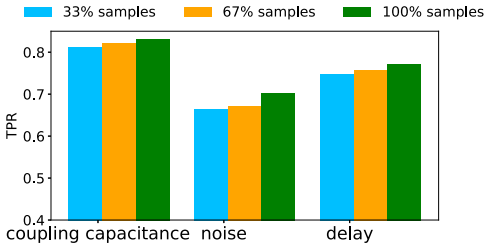

Figure 7: Impact of training sample counts in XGboost.

than the 1.0× threshold by the another. We also investigated the consistency of the XGboost model. From our experimental data we can find that, for coupling capacitance, all the nets predicted positive with the 2× threshold by the XGboost were also predicted positive with the original threshold. In terms of crosstalk-induced noise and delay, among all the nets predicted larger than the 2× threshold, 99.77% and 97.26% were also predicted larger than the original threshold, respectively. This indicates that the XGboost model is stable and consistent.

The number of training samples also has a significant impact on prediction performance. Figure 7 shows how the TPR of XGboost models changes with the number of training samples, at the same FPR. We can observe an about 4% drop in TPR if we only use one third of training samples.

5.6 Runtime and Memory Usage

Given the extracted features illustrated in Section 3.2, the inference time of three XGboost models on an Intel XeonE5-2680 CPU only took 0.003s–0.006s for one placement instance of various sizes in our benchmarks. Compared to the global routing runtime in Table 1, the XGboost models can achieve over 500× speedup. For large industrial designs, the runtime advantage of the proposed method will be more outstanding. Besides, the training of three XGboost models takes only about 10s on a RTX 2080Ti GPU. Moreover, a XGboost model consumes less than 300KB. And the peak runtime memory usage of XGboost-based prediction is about 1GB. As such, the proposed crosstalk prediction approach is very runtime- and memory-efficient.

6 CONCLUSION

In this work, we present a routing-free ML-based crosstalk prediction framework. Given a placement, we extract net topology-related features, along with electrical, logical, and timing-related features. Machine-learning techniques are then employed to train crosstalk prediction models, which classify the nets that are likely to have large coupling capacitance, crosstalk-induced noise, or incremental delay. Experimental validation on 12 benchmark circuits shows that the proposed method can classify more than 70% of crosstalk-critical nets after placement with a FPR of less than 2%. The computation speed is two orders of magnitude faster than a conventional method based on global routing. These results demonstrate that the proposed framework can serve as an accurate early-stage crosstalk evaluation engine that does not require routing information.

ACKNOWLEDGMENTS

This work is partially supported by Semiconductor Research Corporation Tasks 2810.021 and 2810.022 through UT Dallas' Texas Analog Center of Excellence (TxACE).

REFERENCES

[1] Kanak Agarwal, Dennis Sylvester, and David Blaauw. 2006. Modeling and analysis of crosstalk noise in coupled RLC interconnects. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25, 5 (2006), 892–901.
[2] Christoph Albrecht. 2005. IWLS 2005 benchmarks. In *IEEE International Workshop for Logic Synthesis (IWLS)*. 9–9.
[3] Charles J Alpert, Anirudh Devgan, and Stephen T Quay. 1999. Buffer insertion for noise and delay optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 18, 11 (1999), 1633–1645.
[4] Murat R Becer, David Blaauw, Ilan Algor, Rajendran Panda, Chanhee Oh, Vladimir Zolotov, and Ibrahim N Hajj. 2004. Postroute gate sizing for crosstalk noise reduction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 23, 12 (2004), 1670–1677.
[5] Murat R Becer, David Blaauw, Ibrahim N Hajj, and Rajendran Panda. 2002. Early probabilistic noise estimation for capacitively coupled interconnects. In

ACM/IEEE International Workshop on System-level Interconnect Prediction (SLIP). 77–83.

[6] Lauren Hui Chen and Malgorzata Marek-Sadowska. 2002. Incremental delay change due to crosstalk noise. In *ACM International Symposium on Physical Design (ISPD)*. 120–125.

[7] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *ACM International Conference on Knowledge Discovery and Data Mining (DMKD)*. 785–794.

[8] Lawrence T Clark, Vinay Vashishtha, Lucian Shifren, Aditya Gujja, Saurabh Sinha, Brian Cline, Chandarasekaran Ramamurthy, and Greg Yeric. 2016. ASAP7: A 7-nm finFET predictive process design kit. *Microelectronics Journal* 53 (2016), 105–115.

[9] Chunjie Duan, Anup Tirumala, and Sunil P Khatri. 2001. Analysis and avoidance of cross-talk in on-chip buses. In *IEEE International Symposium on High Performance Interconnects (ISHPI)*. 133–138.

[10] Fang-Yu Fan, Hung-Ming Chen, and Isabel Liu. 2010. Technology mapping with crosstalk noise avoidance. In *IEEE Asia and South Pacific Design Automation Conference (ASP-DAC)*. 319–324.

[11] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *IEEE Conference and Workshop on Neural Information Processing Systems (NIPS)*. 1024–1034.

[13] Andrew B Kahng, Mulong Luo, and Siddhartha Nath. 2015. SI for free: machine learning of interconnect coupling delay and transition effects. In *ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*. 1–8.

[14] Taghi M Khoshgoftaar, Moiz Golawala, and Jason Van Hulse. 2007. An empirical study of learning from imbalanced data using random forest. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*. 310–317.

[15] Martin Kuhlmann and Sachin S Sapatnekar. 2001. Exact and efficient crosstalk estimation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 20, 7 (2001), 858–866.

[16] Jinan Lou and Wei Chen. 2004. Crosstalk-aware placement. *IEEE Design & Test of Computers* 21, 1 (2004), 24–32.

[17] Giovanna Menardi and Nicola Torelli. 2014. Training and assessing classification rules with imbalanced data. *Data Mining and Knowledge Discovery* 28, 1 (2014), 92–122.

[18] Phiroze N Parakh and Richard B Brown. 1999. Crosstalk constrained global route embedding. In *ACM Proceedings of the International Symposium on Physical Design (ISPD)*. 201–206.

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. In *IEEE Conference on Neural Information Processing Systems (NIPS)*. 8024–8035.

[20] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[21] Khalid Rahmat, Jose Neves, and Jin-Fuw Lee. 1998. Methods for calculating coupling noise in early design: a comparative analysis. In *IEEE/ACM International Conference on Computer Design (ICCAD)*. 76–81.

[22] Haoxing Ren, David Pan, and Paul G. Villarubia. 2004. True crosstalk aware incremental placement with noise map. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. 402–409.

[23] Hariharan Sankaran and Srinivas Katkoori. 2009. Simultaneous Scheduling, Allocation, Binding, Re-Ordering, and Encoding for Crosstalk Pattern Minimization During High–Level Synthesis. *IEEE Transactions on Very Large Scale Integration Systems* 19, 2 (2009), 217–226.

[24] Peter Spindler and Frank M Johannes. 2007. Fast and accurate routing demand estimation for efficient routability-driven placement. In *IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1–6.

[25] Hsiao-Ping Tseng, Louis Scheffer, and Carl Sechen. 2001. Timing-and crosstalk-driven area routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 20, 4 (2001), 528–544.

[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[27] Ashok Vittal and Malgorzata Marek-Sadowska. 1997. Crosstalk reduction for VLSI. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 16, 3 (1997), 290–298.

[28] Di Wu, Jiang Hu, Rabi Mahapatra, and Min Zhao. 2004. Layer assignment for crosstalk risk minimization. In *IEEE Asia and South Pacific Design Automation Conference (ASP-DAC)*. 159–162.

[29] Zhiyao Xie, Yu-Hung Huang, Guan-Qi Fang, Haoxing Ren, Shao-Yun Fang, Yiran Chen, and Jiang Hu. 2018. RouteNet: Routability prediction for mixed-size designs using convolutional neural network. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–8.

[30] Hai Zhou and Martin D F Wong. 1999. Global routing with crosstalk constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 18, 11 (1999), 1683–1688.