Haoxing Ren
Jiang Hu *Editors*

Machine Learning Applications in Electronic Design Automation

Springer

Chapter 3
Net-Based Machine Learning-Aided Approaches for Timing and Crosstalk Prediction

Rongjian Liang, Zhiyao Xie, Erick Carvajal Barboza, and Jiang Hu

3.1 Introduction

In digital circuit design, timing is a primary design objective that needs to be considered since the very early design stages. Accurate timing prediction is very challenging at early stages due to the absence of information determined by later stages in the design flow. For example, locations of cells and the exact routing topology are critical for timing analysis, but they are not available until cell placement stage and detailed routing stage, respectively, have been executed. However, early design stages have relatively ample room for changes that can fix timing problems in a proactive manner. As a design proceeds to later stages, the design flexibility diminishes although timing estimation becomes more and more accurate. In addition, even at the sign-off stage where all detailed layout parameters are determined, the license cost of EDA tools and the runtime overhead to consider the complex signal integrity (SI) effects hinder the acquisition of accurate timing reports.

In conventional VLSI design flow, two tactics are employed to address the uncertainty in timing analysis. One is to take an overly pessimistic estimation to ensure that no timing violations will occur after routing. However, such pessimism causes overdesign that wastes power, area, and optimization time. The other is to iterate back to early stages, i.e., cell placement or even logic synthesis stage, when the desired timing-power trade-off cannot be achieved at sign-off. However,

R. Liang (✉) · E. C. Barboza · J. Hu
Texas A&M University, College Station, TX, USA
e-mail: liangrj14@tamu.edu; ecarvajal@tamu.edu; jianghu@tamu.edu

Z. Xie
Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong
e-mail: eezhiyao@ust.hk

H. Ren, J. Hu (eds.), *Machine Learning Applications in Electronic Design Automation*, https://doi.org/10.1007/978-3-031-13074-8_3

an additional iteration may not guarantee success, and multiple iterations would grossly increase design turnaround time.

Crosstalk is one of the major threats to SI and timing closure in ASIC design. Through capacitive coupling, a signal switching of one net causes crosstalk noise and incremental delay at its neighboring nets [29]. Noise amplitude can even reach up to 30% of V_{DD} [25], which may exceed the threshold voltage of transistors and lead to glitches, imposing 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 300 ps [26], comparable to clock periods of modern high-performance processors.

Similar to timing analysis, an accurate estimation of crosstalk effects is only possible after detailed routing, where few rooms remain to fix all the crosstalk-induced design problems, even if we identify every crosstalk issue [29]. In this regard, many research efforts have been undertaken to predict and mitigate crosstalk problems at earlier design stages, e.g., placement [20, 26]. However, a majority of crosstalk-driven placement works resort 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.

In recent years, machine learning (ML) techniques have been adopted to improve the predictability of timing and crosstalk effects at different stages of the chip design flow. As for the preplacement stages, ML solutions have been developed for the estimate of the overall wirelength of a netlist [19], lengths of a few selected paths [12], and the net length as well as delay for each individual net [33]. Note that wirelength/net length is an important optimization objective in VLSI design since interconnect is a dominating factor for performance and power in advanced technology nodes. Timing and crosstalk effect estimation at the cell placement stage have been studied in [2] and [17], respectively. ML techniques have also been leveraged to calibrate non-SI timing to SI timing [14], non-SI to non-SI, or SI to SI between different timers [9]. Unlike conventional methods either being very inaccurate or very runtime expensive, ML-aided solutions can be fast yet accurate, reaching a new balance point between runtime and accuracy. Estimation results can essentially benefit design automation by providing early and high-fidelity feedback and guiding proactive actions in early stages to improve design outcomes.

A majority of the aforementioned methods utilize net-based models. Here, net-based models include those modeling the timing or crosstalk properties of individual cell, net, or logic stage. The timing performance of a design can be inferred by propagating cell delays and net delays along the netlist. Crosstalk occurs between physically adjacent nets. Hence, it is natural to build net-based models. Net-based prediction results can be easily integrated into existing EDA flow, since various timing and crosstalk optimization techniques target cells or nets, e.g., gate sizing [6], buffer insertion [11], and layer assignment [30]. Compared with directly modeling path delays or properties of a netlist, net-based estimation provides more detailed information to guide optimization techniques.

In this chapter, we will cover net-based machine learning-aided approaches for timing and crosstalk prediction in detail. A comprehensive review is presented in Sect. 3.2, followed by four representative case studies introduced in Sect. 3.3 to 3.6. Finally, conclusions are drawn in Sect. 3.7.

3.2 Backgrounds on Machine Learning-Aided Timing and Crosstalk Estimation

3.2.1 Timing Prediction Background

Timing analysis determines if the timing constraints imposed on a digital circuit are met. According to whether input vectors are applied, there are two types of timing analysis, i.e., static timing analysis (STA) and dynamic timing analysis. STA checks static timing requirements without applying any input vectors, while dynamic timing analysis applies input vectors and verifies whether output vectors are correct. Compared to dynamic timing analysis, STA is faster as it does not need to simulate multiple input vectors, and it is more thorough since it determines the worst-case timing. All the timing prediction methods introduced in this chapter belong to STA.

We use the example in Fig. 3.1 to illustrate the basic concepts in timing analysis. A timing arc represents the timing relationship between two pins or input/output ports. Timing arcs can be roughly divided into two categories, i.e., net arcs and cell arcs. For example, in Fig. 3.1, the timing arc between Pin c in Net A and Pin y in Net C is a cell arc, while the arc between Pin y and Pin e in Net C is a net arc. The arc from an input pin of a net driver to a sink pin of this net describes the timing for a logic stage. Some previous works develop separate models for net arcs and cell arcs [33], while others build logic stage-based models [2, 14]. For simplicity, a few pre-routing STA methods [2] do not differentiate among rising delays and falling delays. A data path starts from an input port of a design or a clock pin of a sequential cell (e.g., flip-flop/latch/register) and ends at a data input pin of a sequential cell or an output port of the design. For example, in Fig. 3.1, the path from the input pin of Flip-flop F1 to Pin g of Flip-flop F3 is a data path. A clock path starts from the clock input port of the design to a clock pin of a sequential cell. Path delays can be calculated by propagating cell delays and net delays along timing paths. Signal arrival time is the time in which signal arrivals at the pins/ports. Slack is defined as the difference between signal arrival time and the required arrival time. Positive slacks mean that timing constraints are met, while negative slacks imply violations of timing constraints.

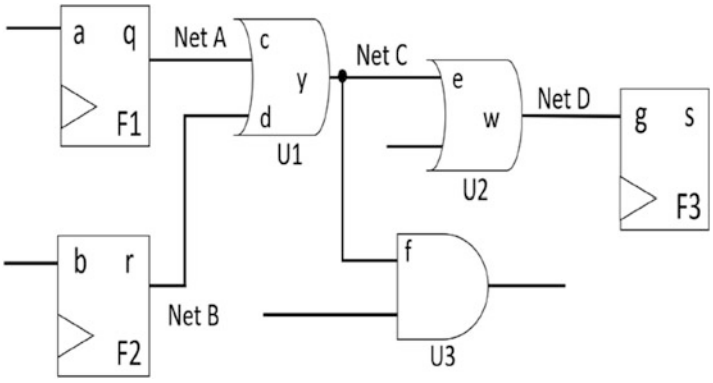

Fig. 3.1 A circuit example [2]

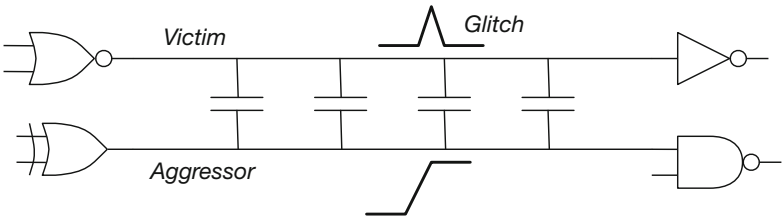

Fig. 3.2 Illustration of crosstalk noise

3.2.2 Crosstalk Prediction Background

Crosstalk refers to the undesirable electrical interaction caused by the capacitive cross-coupling between physically neighboring nets [24]. In advanced technology nodes, metal wires tend to be tall and thin and routed close to each other, leading to increased coupling capacitance between neighboring nets. As shown in Fig. 3.2, due to the coupling capacitance, the rising edge on the aggressor net causes a noise bump or glitch on the victim net, which should be constant at logic 0 or 1. In addition, crosstalk can also lead to signal delays by changing the times at which signal transitions occur, as shown in Fig. 3.3.

3.2.3 Relevant Design Steps

The netlist of a design is generated at logic synthesis stage. It is possible to build net-based models for timing and crosstalk estimation at every major design step since logic synthesis. There is a trade-off between accuracy of timing/crosstalk prediction and design flexibility across relevant design stages. An ideal timing closure and crosstalk avoidance flow can be as follows:

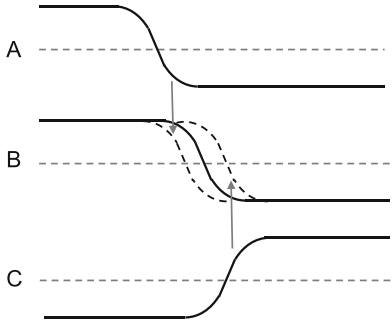

Fig. 3.3 Illustration of transition slowdown or speedup induced by crosstalk [24]

1. During logic synthesis and placement stage, significant timing violations and crosstalk risks are identified and resolved by leveraging the greater design flexibility. Typical optimization techniques at logic synthesis and placement stage include buffer insertion [11], gate sizing [6], and logic restructuring [21].
2. Most of the remaining timing violations and crosstalk risks are eliminated in global routing via optimization techniques, such as buffer insertion, layer assignment [30], and area routing [27].
3. In detailed routing, complete timing closure and crosstalk avoidance is achieved with the precise timing and crosstalk evaluation.

In conventional design flows, there are mainly two kinds of timing models that are often employed for circuit logic and physical synthesis. One is the sign-off model that evaluates if a circuit design satisfies timing specifications. It estimates gate delay using lookup tables [18] or current source model, where slew rate is considered, and wire delays use high-order models [23]. Additionally, sign-off timing analysis tools consider many complicated details such as rising/falling switching, crosstalk, false paths, and simultaneous switching. These models are accurate but very slow. The other is a relatively fast model that is often invoked within synthesis and optimizations. In this case, a gate is modeled as an RC switch and wires are modeled as RC trees, and delay is computed using the Elmore method [7]. Gate and wire delays are collected through addition/subtraction and max/min operations in PERT traversals [4] to obtain signal arrival time, required arrival time, and slack at each node in a circuit. However, such a model is not accurate, as it does not consider potential wire detours due to congestion avoidance and layer assignment impact. Moreover, higher-order interconnect model has no ground to carry out accurate delay computation without wire parasitic information. The aforementioned reasons have motivated recent research in ML-based approaches for fast yet precise timing prediction.

A majority of previous crosstalk avoidance methods target at routing stages [22, 27, 29, 34], the placement stage solutions [20, 26] 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. In addition, timing analysis tools that consider complicated crosstalk effects at sign-off timing analysis are usually very slow. ML-

aided methods for crosstalk prediction have shown great potential in addressing these challenges.

3.2.4 ML Techniques in Net-Based Prediction

Machine learning techniques utilized in net-based prediction can be roughly divided into three categories, i.e., graph neural networks (GNNs), decision tree-based models, and traditional ML techniques. Netlists in fact are graphs; thus, it is natural to apply GNNs for net-based prediction. In decision tree-based models, knowledge is represented by a set of binary decision-making. Such high-level interpretation of knowledge allows decision tree-based models to easily incorporate features from different sources. Traditional ML techniques, such as linear models, multilayer artificial neural networks (ANNs), and support vector machines (SVMs), are also leveraged for timing and crosstalk prediction. We briefly introduce the aforementioned ML techniques in the upcoming paragraphs.

Graph Neural Networks GNN [31] models are composed of multiple sequential convolution layers, as shown in Fig. 3.4. Each layer generates a new embedding for every node based on the previous embeddings. For node n_k with node features O_k, we denote its embedding at the tth layer as $h_k^{(t)}$. Its initial embedding is the node features $h_k^{(0)} = O_k$. In each layer t, GNNs calculate the updated embedding $h_k^{(t)}$ based on the previous embedding of the node itself $h_k^{(t-1)}$ and its neighbors $h_b^{(t-1)} | n_b \in \mathcal{N}(n_k)$.

We show one layer of graph convolutional network (GCN) [16], GSage [8], and graph attention network (GAT) [28] below. Notice that there exist other expressions of these models. The two-dimensional learnable weight at layer t is $W^{(t)}$. In GAT, there is an extra one-dimensional weight $\theta^{(t)}$. The operation [||] concatenates two vectors into one longer vector. Functions σ and g are sigmoid and Leaky ReLU activation function, respectively.

GCN (with self-loops):

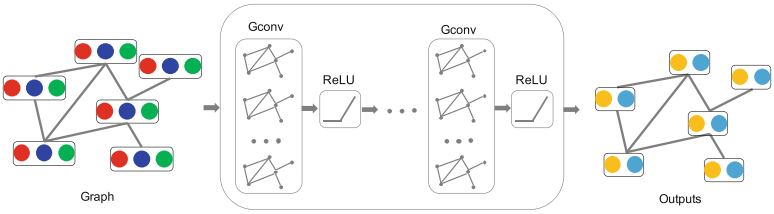

Fig. 3.4 Illustration of the graph neural networks (adapted from [31])

$$h_k^{(t)} = \sigma\left(\sum_{n_\beta \in \mathcal{N}(n_k)\cup\{n_k\}} a_{k\beta} W^{(t)} h_\beta^{(t-1)}\right)$$

$$\text{where } a_{k\beta} = \frac{1}{\sqrt{deg(k)+1}\sqrt{deg(\beta)+1}} \in \mathbb{R}$$

GSage:

$$h_k^{(t)} = \sigma\left(W^{(t)}\left[h_k^{(t-1)} || \frac{1}{deg(k)} \sum_{n_b \in \mathcal{N}(n_k)} h_b^{(t-1)}\right]\right)$$

GAT:

$$h_k^{(t)} = \sigma\left(\sum_{n_\beta \in \mathcal{N}(n_k)\cup\{n_k\}} a_{k\beta} W^{(t)} h_\beta^{(t-1)}\right)$$

$$\text{where } a_{k\beta} = softmax_\beta(r_{k\beta}) \quad \text{over } n_k \text{ and its neighbors,}$$

$$r_{k\beta} = g\left(\theta^{(t)\intercal}\left[W^{(t)} h_\beta^{(t-1)} || W^{(t)} h_k^{(t-1)}\right]\right) \in \mathbb{R}$$

Here, we briefly discuss the difference between these methods. GCN scales the contribution of neighbors by a predetermined coefficient $a_{k\beta}$, depending on the node degree. GSage does not scale neighbors by any factor. In contrast, GAT uses learnable weights W, θ to firstly decide node n_β's contribution $r_{k\beta}$ and then normalize the coefficient $r_{k\beta}$ across n_k and its neighbors through a softmax operation. Such a learnable $a_{k\beta}$ leads to a more flexible model. For all these GNN methods, the last layer's output embedding $h_k^{(t)}$ is connected to a multilayer ANN.

Decision Tree-Based Models Knowledge is represented by a set of binary decision-making in decision tree models. An advantage of these models is that they can ensemble knowledge from different sources due to its high-level interpretation of knowledge and problems. It is especially important for timing and crosstalk prediction since layout, electrical, and logic parameters are all need to be considered. Strong learners can be obtained by ensembling simple decision trees. Random forest [3] and gradient boosted decision trees (GBDT) [5] (shown in Fig. 3.5) are two popular ensemble models. In random forest models, decision trees are used as parallel learners, and each tree is fit to a set of bootstrapping samples taken from the original dataset. Bootstrapping means randomly selecting samples from the original dataset with replacement. The final prediction result is obtained by averaging the results of decision trees. In GBDT models, each decision tree is fit

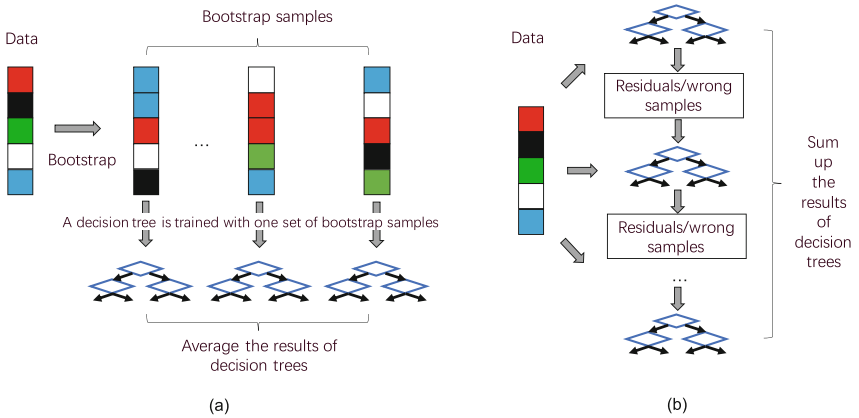

Fig. 3.5 Illustration of the random forest model and the GBDT model. (**a**) Random forest. (**b**) Gradient boosted decision trees

to the residuals from previous ones, and the final result is obtained by summing up the results of all trees.

Traditional ML Techniques Linear models, such as the linear regression and the logistic regression, are also utilized for timing and crosstalk estimation. The main advantage of the linear model is its simplicity. However, it might be too simple for complicated learning tasks.

An ANN consists of multiple fully connected layers and activation layers (e.g., sigmoid and ReLu). A key strength is its ability to capture nonlinear attributes in data [10]. Another advantage is that off-the-shelf neural network engines allow easy customization to the loss function. The main drawback of neural network models is the lack of interpretability.

SVMs find hyperplanes in a high-dimensional feature space that distinctly separates data points from different classes. Key advantages of SVMs are their robustness against noisy data and their effectiveness in high-dimensional spaces.

3.2.5 Why ML for Timing and Crosstalk Prediction

A significant challenge for timing and crosstalk prediction is that they are determined by the complicated joint effects of layout, electrical, and logic parameters. Thanks to its strong data-driven learning capability, ML is a natural good fit for such complex modeling tasks. Many previous ML-based solutions for timing and crosstalk estimation put an emphasis on feature engineering. The extracted features can be roughly divided into four categories, i.e., layout features, electrical features, logic structure features, and timing reports generated by EDA tools. The extracted

features are then fed into ML engines to deliver timing and crosstalk prediction outcomes. The timing reports used as input are generated at early design stages or without considering complicated crosstalk effects. They incorporate information that is captured by EDA tools, but there exists a gap between these reports and the timing outcome at later stages or at a more accurate analysis mode. Other features enable ML-based estimators to further reduce the gap.

We introduce in detail four representative net-based ML-aided solutions for timing and crosstalk prediction in the following sections. The first one targets at the preplacement net length and timing prediction. The second and the third one focus on pre-routing timing and crosstalk effect prediction, respectively. The last one calibrates the non-SI timing to SI timing at sign-off. These four case studies cover several design steps from logic synthesis to sign-off. Since the available information varies at different steps, these case studies utilize different input features. We emphasize the problem formulation, prediction flow, feature engineering, and machine learning engines in the introduction of these case studies.

3.3 Preplacement Net Length and Timing Prediction

3.3.1 Problem Formulation

The work by Xie et al. [33] targets at the preplacement net length and timing prediction. The net length refers to the half perimeter wirelength (HPWL) of the bounding box of the net after placement. It is a key proxy metric for optimizing timing and power. The timing report generated by an industrial timer after placement is used as the ground truth for timing prediction. ML-based preplacement net length and timing prediction contribute to accurate evaluation of timing and power performance of synthesis solutions.

3.3.2 Prediction Flow

Figure 3.6 shows the overall preplacement flow for both individual net size and timing predictions. It is applied before layout and predicts post-placement design objectives. Prediction results can benefit optimization and evaluation for both synthesis and placement. For the net length estimation, a fast version named Net$^{\mathbf{2f}}$ and an accuracy-oriented version named Net$^{\mathbf{2a}}$ are developed. As Fig. 3.6 shows, both versions extract features directly from the netlist, while Net$^{\mathbf{2a}}$ further captures global information by performing clustering on the circuit netlist.

The timing estimator is constructed and applied to directly predict the delay of each individual timing arc, including cell arcs and net arcs. Besides features used by net size prediction, the preplacement timing report from commercial EDA

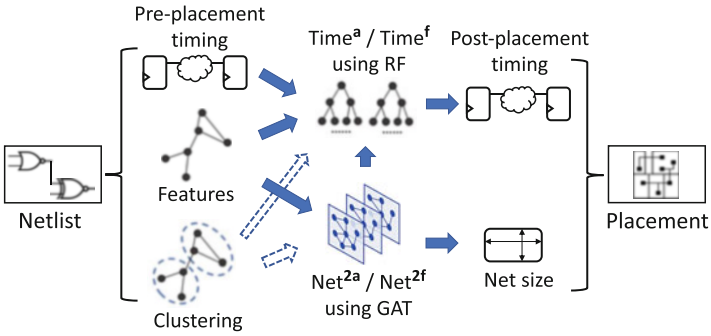

Fig. 3.6 The net size and timing prediction flow [33]

tools is also used as the input. The timing estimators also utilize the information from net size predictions as important input features. Considering the different properties between cell arcs and net arcs, two separate timing prediction models are constructed. Then, based on the inference result, the slack of each circuit node is obtained by traversing the graph with predicted delay values.

3.3.3 Feature Engineering

3.3.3.1 Features for Net Length Prediction

Both global and local topology information are incorporated in [33] for net length and timing prediction. Graph distance between two nodes is evaluated by the number of hops along the shortest path between them. Local information includes the information about the estimated net itself or from its one- to two-hop neighboring nets. In contrast, global information means the pattern behind the topology of the whole netlist or the information from nets far away from the net to be estimated.

The *local* information utilized for net length estimation is shown as follows:

- Physical features: the net's driver area, the sum of areas of all the cells of a net
- Logic structure features: the fan-out size (number of sinks) of the net, the fan-in size (number of input pins of the net's driver cell), the summation and the standard deviation of all neighboring nets' fan-in and fan-out

To capture *global* information, an efficient multilevel partitioning method hMETIS [15] to divide one netlist into multiple clusters/partitions is utilized. The partition method minimizes the overall cut between all clusters and thus provides a global perspective. A few novel global features are extracted based on the clustering results. The most important intuition behind the global features extraction is that,

Table 3.1 Preplacement features for timing prediction [33]

For each cell arc
Preplacement delay of the arc itself
Source pin information: capacitance, slew, slack
All net-size-relevant features of the following net
Predicted size of previous net
For each net arc
Source pin information: max capacitance, slew, slack
Sink pin information: capacitance
All net-size-relevant features of the net
Predicted size of the following net

for a high-quality placement solution, on average, the cells assigned to different clusters tend to be placed far away from each other.

3.3.3.2 Features for Timing Prediction

Table 3.1 summarizes selected features for cell arcs and net arcs. All these features in Table 3.1 are from the three main sources, as summarized below:

- All relevant slew, delay, and slack information from the preplacement timing report.
- Electrical and logic structure information of all relevant nets and cells. It includes the global information captured by performing clustering on the netlist.
- The prediction outcome of Net$^{\mathbf{2f}}$/Net$^{\mathbf{2a}}$.

3.3.4 Machine Learning Engines

3.3.4.1 Machine Learning Engine for Net Length Prediction

To apply graph-based methods, each netlist is converted to one directed graph, as shown in Fig. 3.7. Different from most GNN-based EDA tasks, net length prediction focuses on nets rather than cells. Thus, each net is represented by a node. For each net n_k, it is connected with its fan-ins and fan-outs through their common cells by edges in both directions. The common cell shared by both nets on that edge is called its *edge cell*. For example, in Fig. 3.7b, net n_3 is connected with nets n_4 and n_5 through its sinks c_G and c_H; it is connected with nets n_1 and n_2 through its driver c_D. The edges through edge cell c_G is denoted as $n_3 \rightarrow n_5$ and $n_5 \rightarrow n_3$. The edge cell c_G can also be referred to as c_{35} or c_{53}. Edges in different directions are differentiated by assigning different edge features to $n_3 \rightarrow n_5$ and $n_5 \rightarrow n_3$. After the directed graph is generated, a customized GAT model is applied to the graph to deliver the net length estimation.

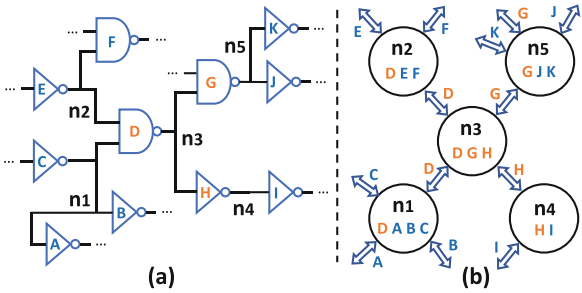

Fig. 3.7 (**a**) Part of a netlist. (**b**) The corresponding graph [33]

3.3.4.2 Machine Learning Engine for Preplacement Timing Prediction

Based on extracted features of the two different types of timing arcs, one cell-arc model and one net-arc model are developed based on the random forest algorithm. Instead of directly predicting the ground truth post-placement delay of each arc, the model by Xie, et al. is actually trained to predict the difference between preplacement and the ground truth post-placement timing. Then, the final predicted delay is the summation of both preplacement delay and the prediction of the incremental delay. This strategy helps the model to directly capture wire-load-induced delay based on the preplacement timing report.

3.4 Pre-Routing Timing Prediction

3.4.1 Problem Formulation

To handle timing uncertainty due to the lack of routing information, designers tend to make very pessimistic predictions, which causes overdesign that wastes chip resources or design effort. To reduce such pessimism, Barboza et al. [2] study the problem of calibrating pre-routing timing to post-routing timing based on ML techniques.

3.4.2 Prediction Flow

Figure 3.8 shows the pre-routing timing prediction flow proposed in [2]. An ML model for predicting the slew of the individual logic stage is first constructed and trained. Besides the pre-routing timing report and the extracted features, the output of the slew model is used to train the model for the prediction of logic stage delay. Then, these models are applied for inference of logic stage delays in PERT traversals [4] of circuit graph in order to obtain arrival time, required arrival time, and slack of each circuit node. The logic stage delay model does not differentiate

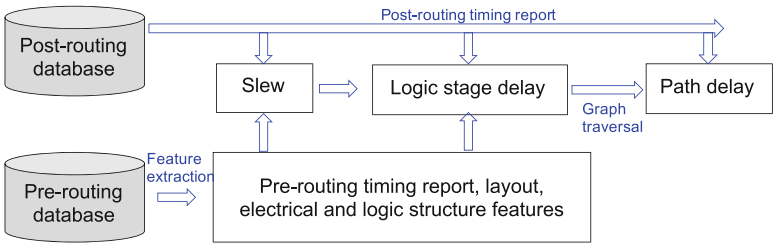

Fig. 3.8 The pre-routing timing prediction flow

among multiple input pins of a driver gate. The different signal arrival times at different input pins are considered during PERT traversal instead of net delay model.

3.4.3 Feature Engineering

Features form input to ML models and their selection is of critical importance for the effectiveness of model application. Net-based delay and slew models share the same features, which are elaborated as follows:

- **Driver and sink capacitance:** Driver output capacitance is generally proportional to its driving strength. Total sink capacitance presents load to the driver. Both of them are the determining factors for net delay and slew. For a net with large capacitive load or on critical paths, buffers may be inserted after placement. The effect of buffer insertion is contained in training data.
- **Distance between the driver and the target sink:** The model predicts delay and slew of one sink at a time, and this sink is called target sink. The horizontal and vertical distances from the driver of the target sink is generally proportional to the corresponding wire delay, especially when buffers are inserted [1].
- **Max driver input slew:** It is obtained using its own ML model, as mentioned above. Here, slew rate is defined to be the signal transition time. Hence, a small slew means sharp signal transition. Both net delay and sink slew are affected by driver input slew. As different types of logic gates may have different number of input pins, we use the maximum slew among all input pins of net driver as feature. This is to accommodate that a machine learning model normally requires fixed input size.
- **Context sink locations:** When a model is applied to estimate the delay/slew of the target sink of a net, the other sinks are called context sinks. For example, consider Net C in Fig. 3.1. When the delay to sink e is estimated, sink f serves as a context sink. Besides contributing to total load capacitance, the locations of context sinks affect routing, buffering, and thus delay/slew at the target sink. Since the number of context sinks varies from one net to another while machine learning model requires input of fixed size, the characteristics of context sink

locations are captured with statistical signatures. One is the median location of all context sinks, which tells roughly how far the context sinks are from the driver. The other is the standard deviations of context sink locations in x-y coordinates. The standard deviation indicates how much the sinks spread out and correlates with the corresponding interconnect tree size as well as the delay/slew at target sink.

3.4.4 Machine Learning Engines

A couple of machine learning engines are constructed and compared in [2], including a linear regression model with L1 regularization, an ANN model, and a random forest model. Experimental results demonstrate that the random forest model achieves the best performance among all the machine learning engines. It is reported that the random forest-based pre-routing timing estimation solution reaches accuracy near post-routing sign-off analysis.

3.5 Pre-Routing Crosstalk Prediction

3.5.1 Problem Formulation

The work by Liang et al. [17] targets three crosstalk classification tasks at placement stage, i.e., identifying

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 3.9 shows the Venn diagram of the above three sets of crosstalk-critical nets. We can find that these sets have overlaps, but they are not identical.

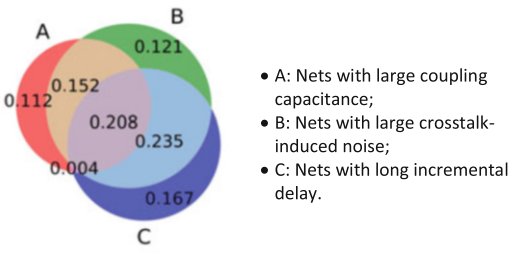

Fig. 3.9 The Venn diagram of three crosstalk-critical net categories [17]. The value on each region shows the proportion of the nets belonging to the region, normalized against the total number of crosstalk-critical nets

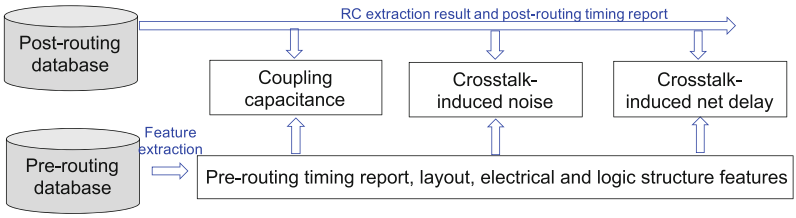

Fig. 3.10 Crosstalk modeling flow

3.5.2 Prediction Flow

The crosstalk modeling flow is shown in Fig. 3.10. Input features and ground truth information are extracted from the placement and the post-routing databases, respectively. By training and evaluating the prediction performance of candidate ML models, e.g., XGBoost, the most effective feature sets and the best models for three crosstalk classification problems are determined, which can be used for fast identification of problematic nets in new placement instances.

The raw input to the crosstalk prediction engine includes placed DEF file, standard cell libraries, and the 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 is generated after detailed routing.

3.5.3 Feature Engineering

3.5.3.1 Probabilistic Congestion Estimation

Routing congestion strongly correlates with crosstalk, since coupling capacitance tends to occur in congested areas. A probabilistic technique for congestion analysis –RUDY (elaborated in Chapter 2)– is employed in [17] due to its great runtime advantage over other techniques and its good correlation with the post-routing solution. Nets can be divided into long-range nets and short-range nets according to their HPWL. In [32], it is shown that routing congestion has a stronger correlation with long-range nets than with shorter ones. In this regard, a longRangeRUDY feature is also extracted by considering only long-range nets when computing the total wire volume.

3.5.3.2 Net Physical Information

Net physical information 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:

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

3.5.3.3 Product of the Wirelength and Congestion

As illustrated in [20], the total coupling capacitance of a net is proportional to the product of its wirelength and the unit coupling capacitance. The product of the HPWL and the RUDY/long-range-RUDY of a net is utilized as an indicator of its coupling capacitance.

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

3.5.3.4 Electrical and Logic Features

The electrical properties of cells play an important role in crosstalk. For example, the crosstalk noise is affected by the driver cell's resistance [26]. Also, the logic type of the driver 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, a logic-based encoding is proposed in [17], 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 counts (e.g., a two-input NAND and a three-input NAND belong to the same group) and different sizes. A variant of the one-hot encoding is utilized to encode the gate groups. Specifically, a vector of length N_g, the total number of gate groups, with only one nonzero entry is used to describe which group the cell belongs to. Unlike assigning 1 to the nonzero entry in the one-hot encoding, the cell's fan-in is assigned to the nonzero 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 driver cell
- **sourceCap**: the output capacitance of the driver cell
- **sinkCap**: the sum of the input capacitance of sink cells
- **fan-in**: the fan-in of the driver cell

3.5.3.5 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 driving strength, consequently more resistant to aggregator's effect in terms of incremental delay.

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

3.5.3.6 Neighboring Net Information

Crosstalk noise and incremental delay depend not only on coupling capacitance but also on the coupling location [26] (near the driver or sink cells) and the aggressors' driving strength. The coupling location is estimated according to the relative location of the net's bounding box and its neighbors' bounding boxes.

- **#Neighboring nets**: the number of neighboring nets
- **#Neighboring long-range nets** : the 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 driver cell and the geometric centers of overlap regions
- **weighted-dist**: the average distance between the driver cell and the geometric centers of overlap regions, weighted by the area of each overlap region
- **dist-source-maxOverlap**: the distance between the driver cell and the geometric center of the largest overlap region

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

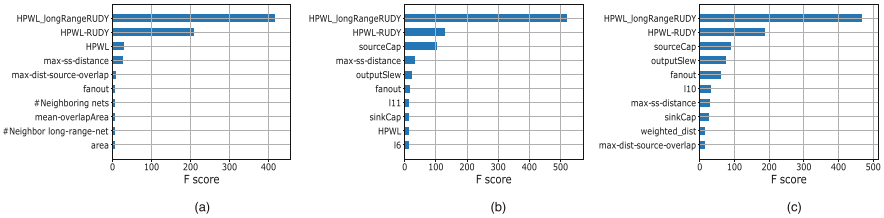

Fig. 3.11 Top 10 important features: (**a**) in coupling capacitance, (**b**) in crosstalk noise, and (**c**) in incremental delay estimations [17]

3.5.4 Machine Learning Engines

Several popular ML techniques, i.e., logistic regression, ANN, random forest, XGBoost, and GNN models, were constructed 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. Experimental results show that the XGBoost method achieves the best performance among all the ML engines.

After training an XGBoost-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 3.11 shows the top 10 important features in coupling capacitance, crosstalk-induced noise, and incremental delay predictions, in terms of "Gain." It can be seen that the layout features play an important role in the three crosstalk estimation tasks because crosstalk heavily depends on layout. As for crosstalk-induced noise and incremental delay prediction, it can be found that electrical features (e.g., sourceCap and sinkCap), logical features (e.g., logic-based encoding for the driver 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.

3.6 Interconnect Coupling Delay and Transition Effect Prediction at Sign-Off

3.6.1 Problem Formulation

The runtime and license costs of SI-enabled timing analysis are typically much larger than those in non-SI mode. The work by Kahng et al. [14] investigates the problem of calibrating sign-off non-SI timing to SI timing with ML techniques. To be specific, the incremental slew, incremental delay due to SI, and SI-aware path

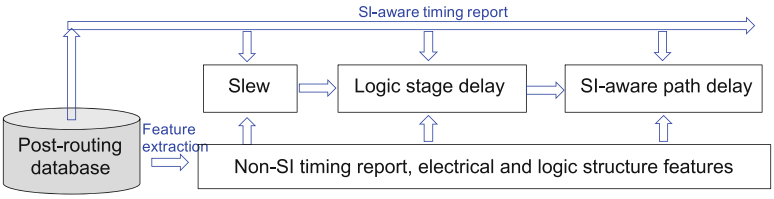

Fig. 3.12 Flow of calibrating non-SI timing to SI timing

delay are estimated given the reports of a sign-off timer that performs only non-SI analysis.

3.6.2 Prediction Flow

Figure 3.12 depicts the flow of calibrating non-SI timing to SI timing. Given a post-routing instance, the non-SI timing report, the electrical and logic structure parameters are extracted as input features. The model for predicting incremental slew due to SI is first constructed and trained, whose outcome is used as input for the training of the SI-induced incremental delay estimator. The predicted delay is utilized for predicting the SI-aware path delay.

3.6.3 Feature Engineering

The following features are extracted for the prediction of SI-aware timing at sign-off:

- Non-SI timing report: slew in non-SI mode, min/max rise/fall delta arrival times between worst aggressor and victim, toggle rate of a victim net, path delay in non-SI mode
- Electrical features: resistance of an arc, coupling capacitance of an arc, ratio of coupling capacitance to the total capacitance, logical effort of the driver cell
- Logic structure: ratio of arc's stage to the total number of stages

It is interesting to note that the work by Kahng et al. does not include layout parameters as input features, since layout information is reflected in parameters such as coupling capacitance, total capacitance, and wire resistance.

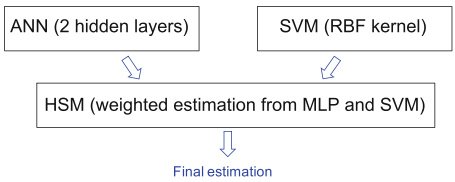

Fig. 3.13 The ensemble model for SI-aware timing estimation (adapted from [14])

3.6.4 Machine Learning Engines

An ensemble model that integrates an ANN and a SVM is leveraged in [14]. The hybrid surrogate modeling (HSM) [13] is utilized to combine the predicted values from the ANN and SVM models and obtain the final estimation, as shown in Fig. 3.13.

3.7 Summary

In this chapter, we introduce net-based ML-aided approaches for timing and crosstalk prediction. Timing and signal integrity are the fundamental objectives in digital circuit design. Conventional timing and crosstalk effect prediction methods are usually either too slow or very inaccurate. Recent ML-aided approaches have demonstrated great potential in providing fast yet accurate prediction. Most of these works build net-based models. After introducing the background on timing and crosstalk modeling as well as relevant ML techniques, we present four representative net-based ML-aided solutions, emphasizing the problem formulation, prediction flow, feature engineering, and ML engines. The key difference between these case studies is in the feature engineering part, since the case studies target different design steps and/or different design properties. There are many other related problems to be studied. A significant one is integrating the ML-based estimators into EDA flows to investigate their impacts on the timing and SI closure.

References

1. Alpert, C.J., Hu, J., Sapatnekar, S.S., Sze, C.: Accurate estimation of global buffer delay within a floorplan. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **25**(6), 1140–1145 (2006)
2. Barboza, E.C., Shukla, N., Chen, Y., Hu, J.: Machine learning-based pre-routing timing prediction with reduced pessimism. In: Proceedings of Design Automation Conference (DAC), pp. 1–6 (2019)
3. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
4. Chang, H., Sapatnekar, S.S.: Statistical timing analysis considering spatial correlations using a single PERT-like traversal. In: Proceedings of International Conference On Computer Aided Design (ICCAD), pp. 621–625 (2003)

5. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining (DMKD), pp. 785–794 (2016)
6. Coudert, O.: Gate sizing for constrained delay/power/area optimization. IEEE Trans. VLSI Syst. **5**(4), 465–472 (1997)
7. Elmore, W.C.: The transient response of damped linear networks with particular regard to wideband amplifiers. J. Appl. Phys. **19**(1), 55–63 (1948)
8. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of the International Conference and Workshop on Neural Information Processing Systems (NIPS), pp. 1024–1034 (2017)
9. Han, S.S., Kahng, A.B., Nath, S., Vydyanathan, A.S.: A deep learning methodology to proliferate golden signoff timing. In: Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1–6 (2014)
10. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. **2**(5), 359–366 (1989)
11. Hu, J., Sapatnekar, S.S.: Simultaneous buffer insertion and non-Hanan optimization for VLSI interconnect under a higher order awe model. In: Proceedings of the International Symposium on Physical Design (ISPD), pp. 133–138 (1999)
12. Hyun, D., Fan, Y., Shin, Y.: Accurate wirelength prediction for placement-aware synthesis through machine learning. In: Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 324–327 (2019)
13. Kahng, A.B., Lin, B., Nath, S.: Enhanced metamodeling techniques for high-dimensional IC design estimation problems. In: Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1861–1866 (2013)
14. Kahng, A.B., Luo, M., Nath, S.: SI for free: machine learning of interconnect coupling delay and transition effects. In: Proceedings of the International Workshop on System Level Interconnect Prediction (SLIP), pp. 1–8 (2015)
15. Karypis, G., Aggarwal, R., Kumar, V., Shekhar, S.: Multilevel hypergraph partitioning: applications in VLSI domain. IEEE Trans. VLSI Syst. **7**(1), 69–79 (1999)
16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks (2016). Preprint. arXiv:1609.02907
17. Liang, R., Xie, Z., Jung, J., Chauha, V., Chen, Y., Hu, J., Xiang, H., Nam, G.J.: Routing-free crosstalk prediction. In: Proceedigs of the International Conference on Computer Aided Design (ICCAD), pp. 1–9 (2020)
18. Lin, S., Lillis, J.P.: Interconnect Analysis and Synthesis. John Wiley & Sons Inc., Hoboken (1999)
19. Liu, Q., Ma, J., Zhang, Q.: Neural network based pre-placement wirelength estimation. In: Proceedings of the International Conference on Field-Programmable Technology (FPT), pp. 16–22 (2012)
20. Lou, J., Chen, W.: Crosstalk-aware placement. IEEE Des. Test Comput. **21**(1), 24–32 (2004)
21. Lou, J., Chen, W., Pedram, M.: Concurrent logic restructuring and placement for timing closure. In: Proceedings of International Conference on Computer-Aided Design (ICCAD), pp. 31–35 (1999)
22. Parakh, P.N., Brown, R.B.: Crosstalk constrained global route embedding. In: Proceedings of the International Symposium on Physical Design (ISPD), pp. 201–206 (1999)
23. Pillage, L.T., Rohrer, R.A.: Asymptotic waveform evaluation for timing analysis. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **9**(4), 352–366 (1990)
24. PrimeTime SI: Crosstalk delay and noise. www.synopsys.com/support/training/signoff/primetimesi-fcd.html (Synopsys)
25. Rahmat, K., Neves, J., Lee, J.F.: Methods for calculating coupling noise in early design: a comparative analysis. In: Proceedings of the International Conference on Computer Aided Design (ICCAD), pp. 76–81 (1998)
26. Ren, H., Pan, D., Villarubia, P.G.: True crosstalk aware incremental placement with noise map. In: Proceedings of the International Conference on Computer Aided Design (ICCAD), pp. 402–409 (2004)

27. Tseng, H.P., Scheffer, L., Sechen, C.: Timing- and crosstalk-driven area routing. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **20**(4), 528–544 (2001)
28. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks (2017). Preprint. arXiv:1710.10903
29. Vittal, A., Marek-Sadowska, M.: Crosstalk reduction for VLSI. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **16**(3), 290–298 (1997)
30. Wu, D., Hu, J., Mahapatra, R., Zhao, M.: Layer assignment for crosstalk risk minimization. In: Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 159–162 (2004)
31. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. **32**(1), 4–24 (2020)
32. Xie, Z., Huang, Y.H., Fang, G.Q., Ren, H., Fang, S.Y., Chen, Y., Hu, J.: RouteNet: routability prediction for mixed-size designs using convolutional neural network. In: Proceedings of the International Conference on Computer-Aided Design (ICCAD), pp. 1–8 (2018)
33. Xie, Z., Liang, R., Xu, X., Hu, J., Duan, Y., Chen, Y.: Net2: a graph attention network method customized for pre-placement net length estimation. In: Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 671–677 (2021)
34. Zhou, H., Wong, M.D.F.: Global routing with crosstalk constraints. IEEE Trans. Comput. Aided Desig. Integr. Circuits Syst. **18**(11), 1683–1688 (1999)