2018 IEEE 36th International Conference on Computer Design

Using Machine Learning to Predict Path-Based Slack from Graph-Based Timing Analysis

Andrew B. Kahng[†‡], Uday Mallappa[‡], Lawrence Saul[†]
[†]CSE and [‡]ECE Departments, UC San Diego, La Jolla, CA, USA
{abk, umallapp, saul}@ucsd.edu

***Abstract*—With diminishing margins in advanced technology nodes, accuracy of timing analysis is a serious concern. Improved accuracy helps to reduce overdesign, particularly in P&R-based optimization and timing closure steps, but comes at the cost of runtime. A major factor in accurate estimation of timing slack, especially for low-voltage corners, is the propagation of transition time. In *graph-based analysis* (GBA), worst-case transition time is propagated through a given gate, independent of the path under analysis, and is hence pessimistic. The timing pessimism results in overdesign and/or inability to fully recover power and area during optimization. In *path-based analysis* (PBA), path-specific transition times are propagated, reducing pessimism. However, PBA incurs severe (4X or more) runtime overheads relative to GBA, and is often avoided in the early stages of physical implementation. With numerous operating corners, use of PBA is even more burdensome. In this paper, we propose a machine learning model, based on *bigrams* of path stages, to predict expensive PBA results from relatively inexpensive GBA results. We identify electrical and structural features of the circuit that affect PBA-GBA divergence with respect to endpoint arrival times. We use GBA and PBA analysis of a given testcase design along with artificially generated timing paths, in combination with a classification and regression tree (CART) approach, to develop a predictive model for PBA-GBA divergence. Empirical studies demonstrate that our model has the potential to substantially reduce pessimism while retaining the lower turnaround time of GBA analysis. For example, a model trained on a post-CTS and tested on a post-route database for the leon3mp design in 28nm FDSOI foundry enablement reduces divergence from true PBA slack (i.e., model prediction divergence, versus GBA divergence) from 9.43ps to 6.06ps (mean absolute error), 26.79ps to 19.72ps (99^{th} percentile error), and 50.78ps to 39.46ps (maximum error).**

I. Introduction

Long runtimes of modern electronic design automation (EDA) tools for designs with over a million instances and many multi-corner multi-mode (MCMM) timing scenarios block quick turnaround time in system-on-chip (SOC) design. A significant portion of runtime is spent on analysis of timing across multiple process, voltage and temperature (PVT) corners. At the same time, accurate, signoff-quality timing analysis is essential during place-and-route and optimization steps, to avoid loops in the flow as well as overdesign that wastes area and power. Tools such as [20] and [21] support graph-based analysis (GBA) and path-based analysis (PBA) modes in static timing analysis (STA), enabling a tradeoff of accuracy versus turnaround time.

In GBA mode, pessimistic transition time is propagated at each node of the timing graph. Figure 1(a) illustrates transition propagation from launch flip-flop L2 to capture flip-flop C1. At instance A1, the worst of its input transition times is propagated from input to output. However, the worst transition time occurs on the pin that is not part of the L2-C1 timing path. Since cell delay estimation is a function of input transition time, the GBA-mode delay calculation for instance A2 is performed using a pessimistic transition time. This pessimism accumulates along the timing path, leading to pessimistic arrival time calculation at the endpoint. Further, the transition time at the endpoint influences the setup requirement of flip-flop C1, adding further pessimism to the reported slack of the timing path.

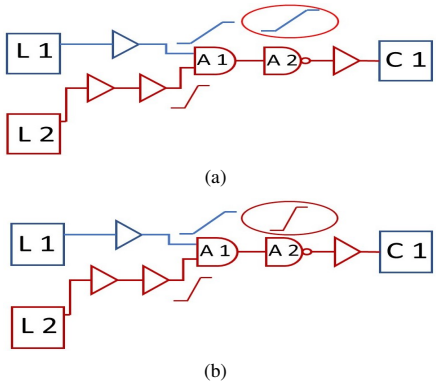


Fig. 1. Transition propagation in (a) GBA mode and (b) PBA mode.

In PBA mode, path-specific transition time is propagated at each node of the timing graph. Figure 1(b) illustrates transition propagation for the L2-C1 timing path in PBA mode. For instance A1, actual path-specific transition time is propagated, and is therefore used in the cell delay calculation for A2. As the number of timing paths to an endpoint increases, there is an exponential increase in possibilities of transition propagation and delay calculation at each node.[1] The path-specific transition propagation and arrival time estimation at each node is runtime-intensive. Figure 2 shows that for public benchmark designs [16] [17] [19] implemented in a 28nm FDSOI foundry enablement, a commercial signoff STA tool exhibits slowdowns (PBA runtime, relative to GBA runtime) as high as 15X for leon3mp [19] (108K flip-flops, 450K signal nets), and 150X for megaboom [17] (350K flip-flops, 960K signal nets).[2] The need for faster path-based analysis is called out in, e.g., Molina [4]; Kahng [5] names prediction of PBA slacks from GBA analysis as a key near-term challenge for machine learning in IC implementation.

Modern IC implementation in advanced nodes relies on MCMM analysis and PBA mode for signoff. Thus, if PBA were to be "fast" (i.e., without significant runtime or other

[1]Details of PBA are given in proprietary tool documentation of major EDA vendors, and analysis outcomes typically have subtle differences across tools.

[2]This slowdown is seen with "`exhaustive`" and "`slack_greater_than -1`" PBA, which assures accuracy in the path-based analysis and provides the least pessimistic basis for optimization. Because runtimes are so long with exhaustive mode, users must typically use "`path`" mode in which path-specific timing recalculation is performed only for some set of timing paths. Analysis using path mode does not guarantee to report worst possible paths to a given endpoint of interest, but can have as little as 2X runtime overhead (`pba_mode path` and `nworst 1`) versus GBA.

2576-6996/18/$31.00 ©2018 IEEE
DOI 10.1109/ICCD.2018.00096

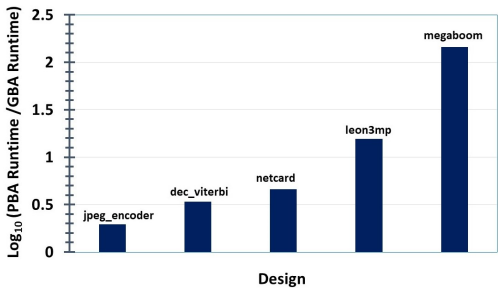


Fig. 2. Ratio of PBA runtime to GBA runtime on log scale (commercial signoff STA tool; 28nm FDSOI foundry enablement) for public-domain design examples (see Table V for details) ranging in size from 4K flip-flops and 40K instances to 350K flip-flops and 990K instances.

overheads) relative to GBA, then only PBA would be used. Unfortunately, today's PBA runtime overheads force the design methodology to make difficult accuracy-runtime tradeoff choices. If there is a high timing violation count in early phases of physical implementation, timing analysis accuracy may not be a primary concern. Hence, designers will typically use less-accurate but relatively inexpensive GBA mode in the early stages of design. Later in the design cycle, as the design converges towards fewer violations, designers must enable PBA mode, at a minimum for timing paths which fail in GBA mode, so as to obtain less-pessimistic, path-specific timing slacks and prevent over-fixing.[3] However, by this time, damage has already been done to the design's power and area metrics as a result of performing GBA-driven optimizations.

Figure 3 illustrates the magnitude of PBA-GBA divergence using a commercial signoff timer for the megaboom testcase implemented in 28nm FDSOI. One worst GBA path is extracted per endpoint (corresponding to "`nworst 1`" in commonly-used STA tool Tcl), and the top 15K timing paths are plotted in decreasing order of PBA-GBA divergence (by the nature of PBA and GBA, the latter is always pessimistic with respect to the former). The maximum PBA-GBA divergence of 110ps means that the GBA can be pessimistic by 110ps as compared to PBA, for this testcase.

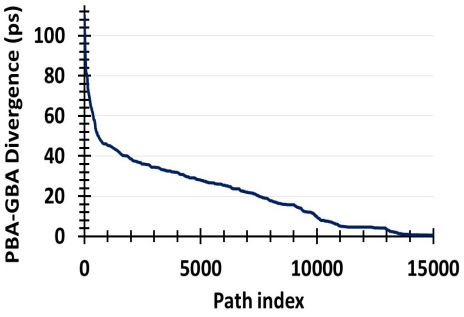


Fig. 3. PBA-GBA divergence for the megaboom design (350K flip-flops, 990K instances) signed off at 1.2ns clock period in 28nm FDSOI technology.

[3]The turnaround time overheads of PBA are compounded by having many MCMM scenarios in timing closure during final stages of implementation, especially for low-power, high-performance designs in advanced nodes.

We emphasize that PBA-GBA divergence is costly – in terms of design quality and/or design schedule – in all contexts. More specifically:

- if GBA slack is positive and PBA slack is positive, then divergence of slack values reduces the ability to exploit available timing slack during power optimization in GBA mode;
- if GBA slack is negative and PBA slack is positive, then the divergence in GBA results in fixing of false violations, with attendant schedule, area and power impacts; and
- if GBA slack is negative and PBA slack is negative, then the divergence in GBA results in over-fixing of timing violations, again impacting schedule, power and area.

Thus, predicting PBA without significant runtime overhead can enable improvement of design quality and schedule, independent of timing slack value. This strongly motivates our present work to develop a fast predictor of PBA from GBA analysis.

In the following, we define *PBA-GBA path arrival time divergence* as the arrival time change at the endpoint of a timing path in PBA mode as compared to GBA mode. We can think of this as a *path-consistent* juxtaposition of the two analyses. Similarly, we define *PBA-GBA transition time divergence* for an arc of a timing path as the delta between transition time in PBA mode as compared to GBA mode. An *endpoint-consistent* definition of divergence is also possible: for endpoint consistency, we define *PBA-GBA endpoint arrival time divergence* as the arrival time difference at a given endpoint between the worst timing path reported by PBA, and the worst timing path reported by GBA. Our work assumes that the clock path does not undergo a significant change in PBA mode as compared to GBA mode. A clock network is primarily comprised of single-input cells, which results in fewer transition propagation conflicts except for rise and fall conflicts. In addition, the sharper (smaller) transition times required in the clock network are more resistant to PBA-GBA variation.[4] With this assumption, we approximate PBA-GBA endpoint slack divergence as PBA-GBA endpoint arrival time divergence. Unless specified, we use the term "PBA-GBA divergence" to indicate PBA-GBA path arrival time divergence.

The major contributions of our work include the following.

- To our knowledge, we are the first to develop a predictor of PBA from GBA, addressing a challenge noted in [4] and [5] and potentially reducing overdesign as well as design schedule.
- We propose a novel combination of *bigram-based* path modeling, classification and regression trees, and selection of model features available from GBA.
- We perform studies in a 28nm FDSOI foundry enablement with a range of open benchmark designs, and demonstrate significant reductions of pessimism, without significant runtime overhead.

The remainder of our paper is organized as follows. In Section II, we introduce terminologies, then elaborate on the background of the PBA-GBA divergence problem before formally defining our problem statement. In Section III, we describe our modeling methodology and model feature selection. In

[4]Our background studies confirm that the change in clock skew in PBA mode is insignificant.

Section IV, we describe our experimental validation setup and results, along with challenges yet unaddressed by our modeling approach. Section V concludes with directions for ongoing and future research.

II. Background

Table I introduces the terminologies and definitions we use in our work.

TABLE I
Terms and definitions.

Term	Definition
Bigram or *bigram unit*	Two consecutive (cell) stages in a timing path
AT	Arrival time
TR	Transition time
PD	Propagation delay
SL	Timing slack
$C_L[j]$	Load capacitance of a driving instance (cell) j
$F_O[j]$	Fanout of driver cell j
$DR[j]$	Drive strength of instance j
N_m	Number of stages in a timing path
$N_s[j]$	Stage depth of instance j relative to launch flop
N_{bg}	Number of bigrams in a timing path
$G[j]$	(Logical) functionality of an instance j
$AT_{gba}[i,j]$ ($AT_{pba}[i,j]$)	AT of instance j, pin i in GBA (PBA) mode
$TR_{gba}[i,j]$ ($TR_{pba}[i,j]$)	TR of instance j, pin i in GBA (PBA) mode
$TR\_MAX_{gba}[j]$	$\max_i\{TR_{gba}[i,j]\}$
$\Delta TR[i,j]$	$TR_{gba}[i,j] - TR_{pba}[i,j]$
$TR\_rat_{gba}[i,j]$	$1 - \frac{TR_{gba}[i,j]}{TR\_MAX_{gba}[i,j]}$ (TR ratio)
$Acc\_TR\_rat_{gba}[i,j]$	$\sum_0^{N_s[j]} TR\_rat_{gba}[i,j]$ (accum. TR ratio)
$PD_{gba}[j]$ ($PD_{pba}[j]$)	PD of an instance j in GBA (PBA) mode
$SL_{gba}[j]$ ($SL_{pba}[j]$)	SL of an endpoint j in GBA (PBA) mode
$\Delta PD[j]$	$PD_{gba}[j] - PD_{pba}[j]$
$\Delta AT[i,j]$	$AT_{gba}[i,j] - AT_{pba}[i,j]$
$\Delta SL[j]$	$SL_{gba}[j] - SL_{pba}[j]$

Figures 4(a) and (b) show a timing path trace with endpoint arrival times of 1.803ns and 1.693ns in GBA and PBA modes, respectively.[5] The PBA-GBA divergence of this timing path is 110ps. Pin U1245606/B has a transition time difference of 38ps and pin U1245606/Z has an arrival time difference of 16ps; these are manifestations of (i) arrival time differences from previous stages and (ii) transition time difference in the current stage due to imbalance in the fanin cone of U1245606. This example highlights the challenge of deciphering graph-based timing analysis reports to estimate hidden path-specific transition time and arrival time information. Section II-C below discusses electrical and physical features that influence the PBA-GBA divergence.

Speed, accuracy and scalability of STA has for decades been a focus of industry R&D attention; see, e.g., TAU Workshop [18] presentations. However, there are few prior works on the use of machine learning for prediction of STA outcomes. Methods to reduce miscorrelation between STA engines or long optimization runtimes are given by [2] and [3], and [6] seeks to reduce STA runtime itself through distributed computing. Kahng et al. [1] provide methodology to model signal integrity (SI) effects on path arrival time, using machine learning. Since their model [1] predicts SI from non-SI, the degree of pessimism in SI prediction is not a primary concern. However, as noted above, model optimism could be a serious concern when predicting PBA from GBA, since an optimistic PBA prediction might mask a real timing violation.

[5]The path report format shown, as well as certain timing analysis option names from tool Tcl mentioned in our discussion, are copyrighted by one or more EDA companies.

A. Problem Statement

Formally, our problem is: Given a training set P_{train} of (PBA-GBA) path-consistent path analysis pairs, such as the pair shown in Figure 4, use P_{train} to train a learning model that predicts PBA-GBA divergence for a testing set P_{test} (where $P_{test} \cap P_{train} = \emptyset$) of paths that are analyzed in only GBA mode. Metrics for evaluation of model quality are described in Section III-A. Experiments used to validate our model are described in Section IV.

B. Intuition for Bigram-Based Modeling

PBA-GBA divergence of a timing path can be estimated either *stage-wise* or *path-wise*. We refer to the latter approach as *lumped* path modeling. For stage-wise modeling, $n \geq 1$ consecutive stages in a timing path are termed an *n-gram* or *n-gram unit* within the path. As n increases, stage-wise modeling (by n-gram units) approaches lumped modeling. The definitions of PBA-GBA divergence in Section I straightforwardly extend to stage-wise modeling. The accumulation of PBA-GBA divergence over the n-grams in a path leads to a path-specific PBA-GBA divergence. Figure 5 shows a bigram-based ($n = 2$) representation of a timing path from launch flip-flop L1 to capture flip-flop C1.

Based on numerous preliminary studies, we have chosen stage bigrams as the fundamental unit for our modeling approach. We observe that lumped modeling shields stage-specific details and inter-stage variations, and in our attempts is prone to large optimistic errors. The lumped approach also has a very large space of features (which in general grows with the number of stages) available to characterize a given path. Furthermore, it is difficult to identify outlier stages that are the "root causes" of misprediction. By contrast, stage-based modeling ensures a fine-grain modeling for each stage and accounts for inter-stage variation of circuit features. Since path prediction is an accumulation of stage predictions, bounding stage-wise errors helps limit path mispredictions. (Practically, we find that it is also easier to identify and diagnose outliers in a stage-based model, and to improve the model by adding features that reduce mispredictions for these outliers.)

We also observe that PBA-GBA divergence arises from the existence of 'competing' transition values at inputs of a cell. However, this will be translated into arrival time divergence only for the *next* stage in a given timing path. This naturally motivates use of a bigram as the basic modeling unit to capture PBA-GBA divergence. We find that the training of n-gram models for $n > 2$ is hampered by the combinatorial explosion of possible n-grams (e.g., over a given cell library), while training of bigram models to achieve accurate prediction is computationally more tractable.

C. Selection of Features

We have evaluated a comprehensive set of electrical and physical features of a bigram unit that can affect PBA-GBA divergence. Our analyses indicate that transition time in GBA mode TR_{gba} at the primary input of a bigram unit, and transition time ratio in GBA mode TR_rat_{gba}, are two mandatory features that strongly impact PBA-GBA divergence of the bigram unit. However, these two features alone are insufficient for accurate prediction of PBA-GBA divergence. Features such as cell drive strength and gate type influence the

Stage	TR	AT INCR	AT
clock clk (rise edge)	0.000000	0.000000	
…			
U1198377/D0 (C12T28SOI_LL_MUXI21X10_P0)	0.046208	0.000065	1.377890
U1198377/Z (C12T28SOI_LL_MUXI21X10_P0)	0.036445	0.032572	1.410461
…			
U1054040/B (C12T28SOI_LL_XNOR2X17_P0)	0.036446	0.000092	1.410553
U1054040/Z (C12T28SOI_LL_XNOR2X17_P0)	0.019427	0.039631	1.450184
…			
U1190225/B (C12T28SOI_LLS_NOR2X55_P0)	0.048106	0.000593	1.552542
U1190225/Z (C12T28SOI_LLS_NOR2X55_P0)	0.028963	0.030666	1.583209
…			
U1245606/B (C12T28SOI_LL_OAI22X15_P10)	0.053207	0.000279	1.690450
U1245606/Z (C12T28SOI_LL_OAI22X15_P10)	0.040024	0.038197	1.728647
U1245373/A (C12T28SOI_LL_NAND2AX27_P10)	0.040025	0.000058	1.728705
U1245373/Z (C12T28SOI_LL_NAND2AX27_P10)	0.020709	0.033629	1.762334
…			
U1244873/B (C12T28SOI_LLBRODS_NAND2X14_P16)	0.038315	0.000074	1.777529
U1244873/Z (C12T28SOI_LLBRODS_NAND2X14_P16)	0.021030	0.026206	1.803736
…			
data arrival time			1.803783

(a)

Stage	TR	AT INCR	AT
clock clk (rise edge)	0.000000	0.000000	
…			
U1198377/D0 (C12T28SOI_LL_MUXI21X10_P0)	0.021821	0.000065	1.363336
U1198377/Z (C12T28SOI_LL_MUXI21X10_P0)	0.024857	0.024075	1.387411
…			
U1054040/B (C12T28SOI_LL_XNOR2X17_P0)	0.024859	0.000092	1.387503
U1054040/Z (C12T28SOI_LL_XNOR2X17_P0)	0.013953	0.034927	1.422430
…			
U1190225/B (C12T28SOI_LLS_NOR2X55_P0)	0.016866	0.000593	1.514668
U1190225/Z (C12T28SOI_LLS_NOR2X55_P0)	0.023576	0.019043	1.533711
…			
U1245606/B (C12T28SOI_LL_OAI22X15_P10)	0.015089	0.000279	1.616198
U1245606/Z (C12T28SOI_LL_OAI22X15_P10)	0.026354	0.022153	1.638351
…			
U1245373/A (C12T28SOI_LL_NAND2AX27_P10)	0.026355	0.000058	1.638409
U1245373/Z (C12T28SOI_LL_NAND2AX27_P10)	0.013768	0.028074	1.666482
…			
U1244873/B (C12T28SOI_LLBRODS_NAND2X14_P16)	0.012127	0.000074	1.679381
U1244873/Z (C12T28SOI_LLBRODS_NAND2X14_P16)	0.011745	0.014285	1.693667
…			
data arrival time			1.693714

(b)

Fig. 4. PBA-GBA divergence for the megaboom design (990K instances) signed off at 1.2ns in 28nm FDSOI technology. Shown: timing analysis for the same path in (a) GBA mode and (b) PBA mode.

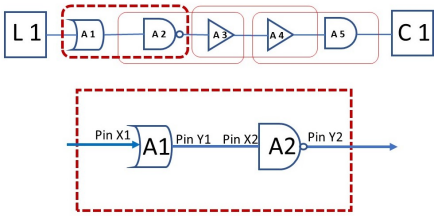
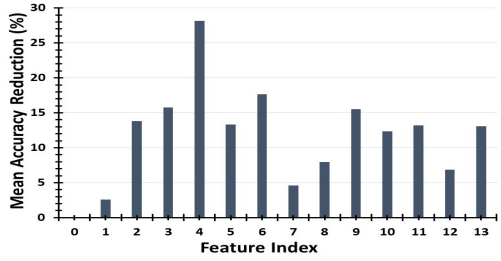


Fig. 5. Bigram-based model of a timing path. The timing path is represented as a series of four bigram units.

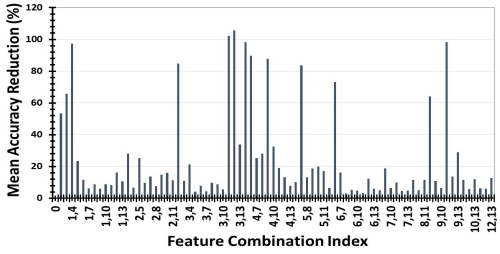
transition time variation at the input cell which is reflected at the bigram output pin. In addition, arrival time of the bigram unit is an indicator of the positioning of the bigram unit along the timing path: the arrival time reflects topological distance from the launch flip-flop, as well as parametric on-chip variation (POCV) derating and error propagation along the timing path. Other layout-dependent electrical features such as output load capacitance and fanouts of the cells in the bigram unit are also found to be useful in predicting PBA-GBA divergence. In total, the bigram-based model for which we report results below uses the following 13 features extracted from GBA analysis:

1) transition time of the first cell in the bigram unit;
2) transition time of the second cell in the bigram unit;
3) arrival time of first cell in bigram unit;
4) transition time ratio (TR) of first cell in bigram unit;
5) arrival time of second cell in bigram unit;
6) drive strength of first cell in bigram unit;
7) drive strength of second cell in bigram unit;
8) functionality of first cell in bigram unit;
9) functionality of second cell in bigram unit;
10) fanout of first cell in bigram unit;
11) load capacitance of first cell in bigram unit;
12) accumulated transition time ratio of first cell in bigram unit;
13) propagation delay of second cell in bigram unit.

Our studies indicate that dropping any one of these features reduces absolute model accuracy by at least 2%, and dropping any two features at a time reduces the model accuracy by at least 4%. Here, we define the *mean_initial* accuracy as the mean of absolute (Predicted − Actual) arrival times, with all 13 features used. The *mean_reduced* accuracy is the mean of absolute (Predicted_New − Actual) arrival times, with reduced features. Then, we define the *model accuracy reduction* (%) as (mean_reduced − mean_initial) × 100 / (mean_initial). Figures 6(a) and (b) illustrate the sensitivity of accuracy reduction to particular features. Accuracy with none of the features dropped is used as baseline for comparison. Dropping TR_rat_{gba} alone reduces the model accuracy by 27%, and dropping any pair combination that includes TR_rat_{gba} corresponds to the largest accuracy reductions in Figure 6(b).



(a)



(b)

Fig. 6. Impact of dropping any of the 13 features on model accuracy. (a) Dropping any single one of the 13 features (indexed as above); the peak loss of accuracy corresponds to TR_ret_{gba}. (b) Dropping any pair of the 13 features at a time; the x-axis gives, from left to right, C(13,2) = 78 pairs (1,2), (1,3), ..., (12,13).

Last, since PBA-GBA divergence depends on the incremental transition time for each bigram unit, we find that it is necessary to implement a *two-phase* modeling strategy: (i) Phase 1 predicts incremental transition time gain in PBA mode, and (ii) Phase 2 uses predictions from Phase 1 along with other features from GBA mode analysis to predict PBA-GBA divergence for the bigram unit. We give more details of this two-phase strategy in the next section.

III. Modeling Methodology

After selection of features, our modeling methodology includes application of machine learning techniques that capture complex interactions of the features and their impact on PBA-GBA divergence. We find that linear regression techniques fail to capture nonlinearity of predictions and complex interactions between features. For example, interaction of features such as input transition, output load and cell drive strength influence PBA-GBA divergence. We have also evaluated nonlinear modeling techniques such as multivariate adaptive regression splines (MARS) [10] which suffer from two-sided distribution of error. Since PBA is always optimistic as compared to GBA, a pessimistic prediction (i.e., prediction of less timing slack than the given GBA slack) is incorrect. With bigram-based modeling, as the number of data points used for modeling increases (1M+), our results indicate that MARS is not scalable when higher-order effects are introduced. Ultimately, for improved accuracy, reduced variance and faster runtimes, we have focused our efforts on tree ensemble methods. Random forests of classification and regression trees give the best results so far, and Figure 7 illustrates the visual aid inherent in tree-based modeling, which helps to better understand feature importance and classification criteria. We discuss more about classification and regression trees in Section III-B.

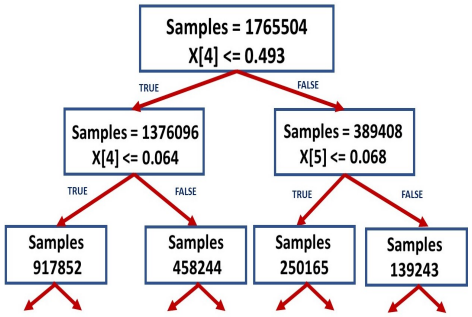


Fig. 7. Tree-based classification with 1.7M training samples and 13 features. Feature X[4], which corresponds to TR_rat_{gba}, splits the data space into 75% and 25% with a split value of 0.493, indicating its importance in classifying input data.

A. Reporting Metrics

PBA-GBA divergence signifies the pessimism in GBA mode. Therefore, reduction in this pessimism is an appropriate metric to evaluate the predictive model. Figure 9 shows a path-consistent plot of actual GBA versus PBA path arrival times. The maximum PBA-GBA divergence is 110ps. The blue band signifies the pessimism in GBA mode as compared to PBA mode. The intent of machine learning-based PBA prediction is to reduce width of the blue band in a predicted PBA versus actual PBA plot. Ideally, the plot of predicted PBA versus actual PBA would be the straight orange line $Y = X$, i.e., zero pessimism in the prediction.

Table II shows actual PBA-GBA divergence metrics from a commercial signoff timer that we use as the reference to quantify the accuracy of our predictive model.

Table III explains path-consistent and endpoint-consistent divergence metrics that we define for model predictions.

TABLE II
PBA-GBA divergence metrics.

Notation	Meaning
$actual\_max_{path}$	Upper bound of actual PBA-GBA divergence
$actual\_99p_{path}$	99^{th} percentile value of sorted PBA-GBA divergence (in ascending order)
$actual\_mean_{path}$	Mean absolute value of actual PBA-GBA divergence

These "model_*" metrics help assess the divergence of model-predicted PBA timing from actual PBA timing. Metrics that indicate our model accuracy are 99^{th} percentile value of divergence, mean absolute value of divergence, and worst-case prediction divergence. Reduction of "model_*" metrics, as compared to reference PBA-GBA divergence "actual_*" metrics, shows reduction of pessimism. This is conceptually portrayed in Figure 8.

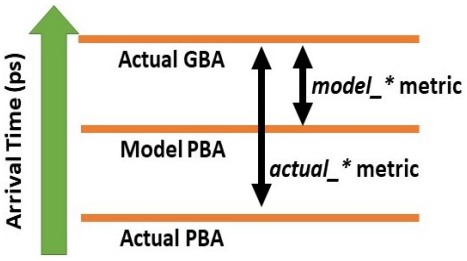


Fig. 8. Reduction of model_* metrics as compared to actual_* divergence metrics signifies reduction of pessimism.

TABLE III
Model prediction-based divergence metrics.

Notation	Meaning
Path-consistent prediction metrics	
$model\_max_{path}$	Worst-case pessimistic prediction divergence
$model\_opt_{path}$	Worst-case optimistic prediction divergence
$model\_99p_{path}$	99^{th} percentile value of absolute prediction divergence values (in ascending order)
$model\_mean_{path}$	Mean absolute value of prediction divergence values
Endpoint-consistent prediction metrics	
$model\_max_{end}$	Worst-case pessimistic prediction divergence
$model\_opt_{end}$	Worst-case optimistic prediction divergence
$model\_99p_{end}$	99^{th} percentile value of absolute prediction divergence values (in ascending order)
$model\_mean_{end}$	Mean absolute value of prediction divergence values

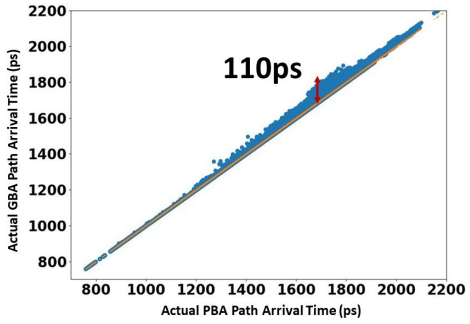


Fig. 9. PBA versus GBA path-consistent arrival times (with a maximum PBA-GBA divergence of $110ps$) reported by a commercial timer for the megaboom testcase in 28nm FDSOI technology.

B. Classification and regression trees

Classification and regression trees (CART) [8] are nonlinear techniques for constructing predictive models from data. The models are obtained by recursively partitioning the data space into feature space and fitting a simple predictive model within each partition. This recursive partitioning can model a data set with complex feature interactions. In the context of PBA-GBA prediction, *regression trees* can be used to predict PBA arrival time for each bigram unit, i.e., we use features of the test data (GBA analysis results) to predict PBA arrival time for each data point. *Classification trees* can predict PBA-GBA divergence (incremental arrival time gain) where the model uses features of the test data to predict PBA-GBA divergence for each data point.

An important realization is that since regression tree-based modeling is limited by the span of PBA arrival time values in the training data, testing is always constrained by the range of arrival time values covered in the training phase. On the other hand, classification tree-based modeling is limited by the range of PBA-GBA arrival time increments used in training data. During testing, if a data point exceeds the class value used in training data, the model is constrained by the span of increments in the training data.

As an example, consider a training data set with GBA and PBA arrival time ranges of 24ps to 345ps, and 14ps to 326ps, respectively, along with PBA-GBA divergence range of 0ps to 40ps. In regression-based modeling, a test data point with GBA arrival time of 560ps is constrained by the span of training data, which is 345ps in this case. In classification-based modeling, predicted PBA-GBA divergence will be from one of the values in the range of 0ps to 40ps. If we ensure that the span of possible PBA-GBA divergence values are covered in the training data, mispredictions can be reduced. In addition, having positive class values gives us "sensibility by construction" in our predictions, since actual PBA-GBA divergence can never be negative.

Our preliminary studies, summarized in Table IV for the netcard testcase, lead us to use the classification tree approach for PBA-GBA divergence prediction.

TABLE IV
REGRESSION VERSUS CLASSIFICATION TREES (NETCARD).

Metric	Regression	Classification
$model\_opt_{path}$	18.65ps	8.21ps
$model\_99p_{path}$	12.96ps	6.44ps

C. Model Definition

For Phase 1 and Phase 2 of our modeling, Equations (1) and (2) capture PBA-GBA divergence in transition time and arrival time respectively, for each bigram unit.

$$\Delta TR_{bg} = f(C_L, DR, G, F_O, TR_{gba}, AT_{gba}, TR\_rat_{gba}, Acc\_TR\_rat_{gba}) \quad (1)$$

$$\Delta AT_{bg} = f(C_L, DR, G, F_O, TR_{gba}, AT_{gba}, TR\_rat_{gba}, Acc\_TR\_rat_{gba}, \Delta TR_{bg}) \quad (2)$$

PBA-GBA divergence for a timing path is estimated by cumulative addition of bigram PBA-GBA divergence values in the timing path. This is explained in Equation (3).

$$\Delta AT_{path} = \sum_{1}^{N_{bg}} \Delta AT_{bg} \quad (3)$$

Our model is an ensemble of 50 regression trees.[6] We use mean squared error as our criterion for tree splitting, and all 13 input features are used in determining a given split point. We do not set any bound on the depth of regression tree, or on the number of leaf nodes for each tree in the ensemble.

D. Modeling Flow

We propose a modeling flow as illustrated in Figure 10. During the model training, both GBA and PBA path-consistent timing reports are used as inputs. We then extract features required to model PBA-GBA divergence for each bigram pair. During the model testing, the model predicts PBA-GBA divergence for any unseen (i.e., new) GBA timing path. Predicted PBA timing results that are output by our model can subsequently serve as, e.g., inputs to optimization and sizing steps of the physical implementation flow.

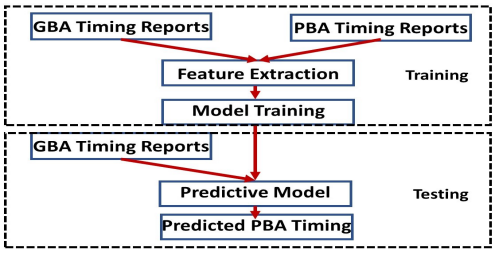


Fig. 10. Our modeling flow.

IV. EXPERIMENTAL VALIDATION

We now describe our design of experiments to validate the predictive model. For each of these experiments, we discuss our modeling results.

A. Design of Experiments

Our experiments use in-house developed artificial designs and five real designs as listed in Table V. Artificial designs are created for potential availability during an initial, "bootstrap" training phase of modeling. We use 28nm FDSOI foundry technology libraries for all our experiments. Training and test data is split using a random number generator. In our experience, the choice of random seed does not significantly impact model accuracy. Runtimes on a single thread on an Intel Xeon 2.6 GHz server are as follows. For a training data set with 2.44M bigrams, data preparation time is 362 seconds, and model training time is 219 seconds; this is a one-time overhead for realistic use cases. After a trained model is available, data preparation time for a test data set with 1.04M bigrams is 178 seconds, and model inference runtime is 17 seconds. For the same testcase, GBA runtime on test data is 156 seconds, whereas PBA runtime is 1080 seconds. For a design going through optimization phases, recurrent PBA

TABLE V
DESIGN DATA USED FOR EXPERIMENTS.

Design	# Instances	# Flip-Flops	# Bigrams
artificial	2.4M	400K	1.3M
megaboom	990K	350K	3.4M
leon3mp	450K	100K	1.8M
netcard	303K	66K	856K
dec_viterbi	61K	26K	200K
jpeg_encoder	40K	4K	60K

runtime costs can be avoided, with additional time invested only for GBA analysis, data preparation, and model inference.

We conduct three experiments to demonstrate accuracy and robustness of our predictive model. We also propose a fourth experiment to generate endpoint-consistent PBA-GBA divergence.

- **Experiment 1 (Accuracy)**: The goal of this experiment is to validate our modeling accuracy. Model is trained with 70% data points of a real design, and tested on unseen 30% data points of the same design.
- **Experiment 2 (Robustness)**: The goal of this experiment is to validate our modeling robustness. Model is trained with data points from the post-CTS database of a real design, and tested on an unseen post-routed implementation of the same design.
- **Experiment 3 (Robustness)**: The goal of this experiment is to validate the span of our artificial testcase development methodology. Model is trained with artificially generated testcases along with a sample of data points (30%) from a real design, and tested on unseen 70% data points of the same design.
- **Experiment 4 (Endpoint Slack)**: The goal of this experiment is to translate path-consistent PBA-GBA divergence predictions to endpoint-consistent PBA-GBA divergence values.

B. Results

In our results, we first compare model-predicted PBA arrival time values with reference PBA results using a commercial timer, while maintaining path consistency. Reduction of model prediction divergence metrics as compared to reference divergence metrics signifies the reduction of PBA-GBA divergence and availability of timing slack for design optimization.

Results of Experiment 1. We use 70% of the timing paths for training and test on 30% of the timing paths of the same design. Figure 12 illustrates the results for Experiment 1. For this and other experiments, we see that our modeling offers larger benefits for larger designs. The data in Table VI shows that for the three largest testcases, the mean, 99^{th} percentile and max divergence metrics reduce by at least 61.7%, 15.9% and 47.5%, respectively, as compared to reference divergence metrics.

Results of Experiment 2. We use timing reports from a post-CTS database as input for training, and test the model on a post-routed database of the same design. This model is particularly helpful to set realistic optimization criteria during routing.

[6]We have evaluated modeling with various numbers of regression trees (1, 10, 25, 50, 75), and find the value of 50 to provide the best tradeoff between accuracy, robustness and runtime.

TABLE VI
MODEL DIVERGENCE IMPROVEMENT IN EXPERIMENT 1.

	Mean		99p		Max	
Design	actual	model	actual	model	actual	model
megaboom	2.59ps	0.99ps	43.95ps	23.05ps	110ps	78ps
leon3mp	10.55ps	2.14ps	30.29ps	7.45ps	50.78ps	42.70ps
netcard	6.70ps	1.52ps	22.62ps	8.52ps	39.59ps	19.90ps
dec_viterbi	0.09ps	0.05ps	3.02ps	1.09ps	21.46ps	20.96ps
jpeg_encoder	3.35ps	2.01ps	16.35ps	12.79ps	27.29ps	26.79ps

Figure 13 illustrates the results for Experiment 2. The data in Table VII show that for the three largest testcases, the mean, 99^{th} percentile and max model divergence metrics reduce by at least 26.6%, 11.7% and 26.3%, respectively, as compared to reference divergence metrics.

TABLE VII
MODEL DIVERGENCE IMPROVEMENT IN EXPERIMENT 2.

	Mean		99p		Max	
Design	actual	model	actual	model	actual	model
megaboom	4.51ps	3.31ps	53ps	36.62ps	119.24ps	89.65ps
leon3mp	9.43ps	6.06ps	26.79ps	19.72ps	50.78ps	39.46ps
netcard	4.29ps	2.49ps	17.20ps	9.78ps	33.56ps	29.63ps
dec_viterbi	0.39ps	0.29ps	10.53ps	8.65ps	22.97ps	21.46ps
jpeg_encoder	2.99ps	1.90ps	17.70ps	12.34ps	27.11ps	21.59ps

Results of Experiment 3.

We use in-house developed artificial designs and a sample from a real design (30% data points) for training, and predict PBA-GBA divergence on the same real design (70% data points). This reflects a hypothetical "ideal scenario", wherein we have the capability to produce artificial testcases that span the entire space of real designs (see Subsection IV-C below). Figure 14 illustrates results from Experiment 3. The data in Table VIII show that for the three largest testcases, the mean, 99^{th} percentile and max model divergence metrics reduce by at least 27.1%, 13.4% and 13.5%, respectively, as compared to reference divergence metrics.

TABLE VIII
MODEL DIVERGENCE IMPROVEMENT IN EXPERIMENT 3.

	Mean		99p		Max	
Design	actual	model	actual	model	actual	model
megaboom	3.06ps	2.23ps	41.14ps	31.04ps	119.24ps	103.21ps
leon3mp	9.07ps	4.50ps	22.59ps	19.55ps	46.46ps	33.96ps
netcard	7.21ps	2.78ps	21.84ps	9.58ps	46.25ps	31.24ps
dec_viterbi	0.41ps	0.29ps	9.98ps	8.09ps	19.74ps	18.67ps
jpeg_encoder	4.75ps	3.98ps	16.51ps	14.63ps	26.47ps	24.54ps

Results of Experiment 4. The goal of this experiment is to translate path-consistent predictions from the model to *endpoint-consistent predictions*. This is done by choosing `nworst` arrival time predictions for each endpoint. We then compare these endpoint-consistent PBA-GBA divergence values to actual endpoint-consistent PBA-GBA divergence values. Figure 11 demonstrates the effect of `nworst` on endpoint-based divergence metrics. As evident from the plots, though the model divergence metrics drop slightly for initial increase of `nworst` value, divergence metrics rise and saturate with further increase of `nworst`. The initial drop in divergence is due to broader coverage of timing paths to an endpoint. However, model error effects increase with inclusion of more

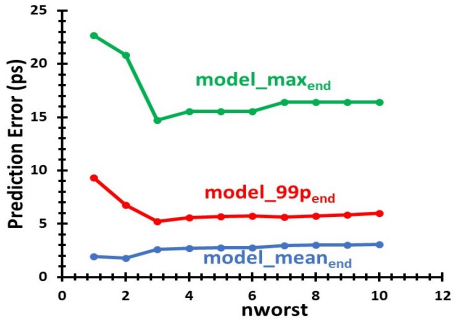


Fig. 11. Plots of prediction error with `nworst` of endpoints.

data points (i.e., worst GBA paths) per endpoint, and the endpoint-consistent accuracy metric saturates with increase in `nworst`. Potentially, `nworst 3` could be a point of interest to derive endpoint-based arrival time or slack predictions. At the same time, achieving better use of, say, `nworst 20` GBA analysis is an important direction for future work.

C. Challenges

Two important remaining challenges for our modeling approach are (i) the reduction or elimination of remaining optimism in PBA slack prediction, and (ii) endpoint-consistent pessimism reduction. In this subsection, we provide several comments regarding the former challenge.

First, while optimistic predictions are evident in our experimental results, we note that for any endpoint, the arrival time gain lower-bounds the slack gain. In other words, for any endpoint, slack gain is likely to be larger than arrival time gain. This is because sharper transition times at the endpoint in PBA mode lead to reduction of the setup time requirement at the endpoint. Since our present model predicts arrival time gain, there is actually some leftover "budget" with respect to slack gain prediction – and this in effect reduces the optimism of our model. For the leon3mp testcase, this "budget" averages 4.46ps over all endpoints, as plotted in Figure 15.

Second, we observe that misprediction is at least partly a consequence of a test data point's distance from nearest training data points in the modeling feature space. In an ideal scenario, our artificial circuits for any (technology and library) design enablement would effectively span (cover) the entire real design space, such that a one-time trained model could accurately predict PBA timing from GBA timing on any real design. However, our current artificial circuits methodology is far from enabling such an ideal use case. Thus, an important direction for future work is to incrementally train models with real design data along with artificial and previous circuit data, always testing on subsequent design iterations. In an extension of our current approach, test data points encountered that are far from training data set could be incrementally included into the training data set to help reduce model mispredictions. Further, detailed PBA analysis can be performed every few design iterations, to identify mispredicted outliers for inclusion in future (incremental) training. Such a methodology might follow the flow shown in Figure 16.[7] Last, we recall the primary motivation of reducing overdesign during the optimization flow – not replacing the golden signoff tool and PBA signoff analysis. The model calibration flow illustrated in Figure 16 helps make isolated optimistic predictions less significant as compared to the benefits obtained from reducing overdesign earlier in the design process.

V. Conclusions

In this work, we are the first to apply machine learning techniques to model PBA-GBA divergence in endpoint arrival times, addressing an important accuracy-runtime tradeoff in static timing analysis [4] [5]. We propose a model based on decision trees along with electrical and physical features of *stage bigrams* in timing paths. We assess potential benefits of our model using 28nm FDSOI foundry technology, a leading commercial signoff STA tool, and implementations of public testcase designs up to 1M+ instances. We measure the decrease of PBA-GBA divergence obtained by the model, according to several metrics and in several usage scenarios. In our experiments, model-predicted PBA arrival times reduce mean, 99^{th} percentile and max divergence metrics by at least 26.6%, 13.4% and 11.7%, respectively as compared to reference PBA-GBA divergence metrics. Such reductions can help avoid overfixing and achieve improved power and area outcomes during optimization. In addition, both model training and inference are efficient, with a training time of 219 seconds and inference time of 17 seconds for a test data set with 1.04M bigrams.

A number of ongoing and future works remain. (1) We are seeking to integrate our predictive models with an academic sizer and optimizer, to explore the benefit from reduced pessimism in MCMM timing closure and sizing for leakage and total power reduction. (2) As shown by Experiment 3 and as discussed in Subsection IV-C, significant work remains to be done toward design of artificial testcases that can train well-performing models for a given design enablement, without reliance on any actual designs. (3) Reduction or elimination of remaining optimism in PBA slack prediction, as well as endpoint-consistent pessimism reduction, present additional challenges for future research. Both the available "budget" of arrival time gain versus slack gain, and the proposed incremental model calibration flow, may provide mitigations for the problem of optimism. (4) Last, as noted in the Experiment 4 discussion, achieving better use of multiple (`nworst` $\gg 1$) GBA paths to a given endpoint is an important direction to pursue.

Acknowledgments

We thank Dr. Tuck-Boon Chan of Qualcomm and Dr. Siddhartha Nath for providing valuable feedback. Research in UC San Diego ABKGroup is supported in part by funding from NSF, DARPA, Qualcomm, Samsung, NXP, Mentor Graphics and the C-DEN center.

[7] A comment: As a design undergoes changes during the design cycle, performing PBA analysis after any given design change would bring significant runtime overhead. The ultimate bar for value obtained from GBA-based predictive model is faster design convergence through avoidance of PBA analysis overheads.

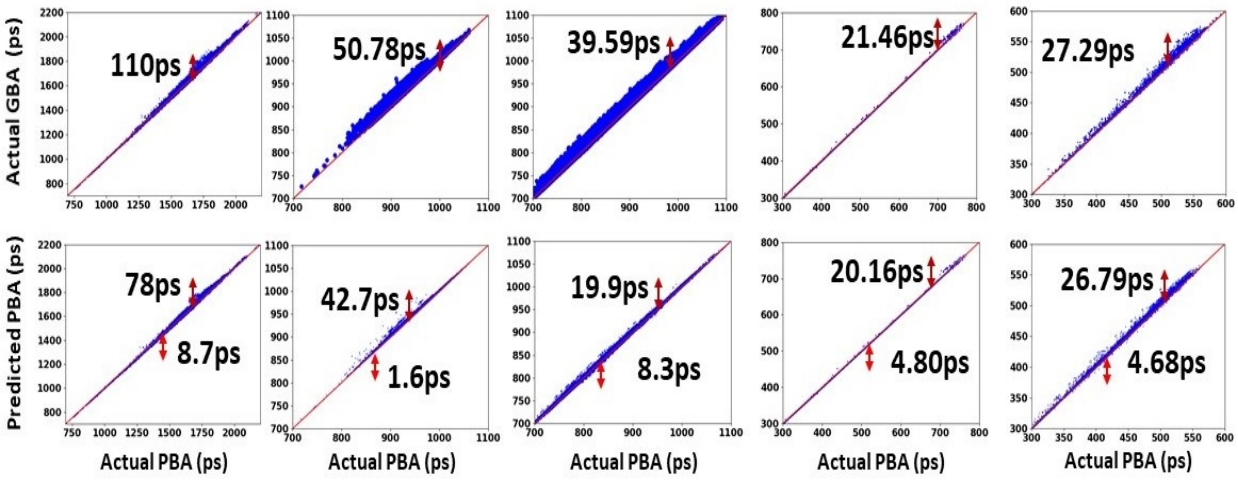


Fig. 12. Results of Experiment 1 for actual GBA path arrival time (top row) and predicted PBA path arrival time (bottom row) versus actual PBA path arrival time for, in left-to-right order, megaboom, leon3mp, netcard, dec_viterbi and jpeg_encoder.

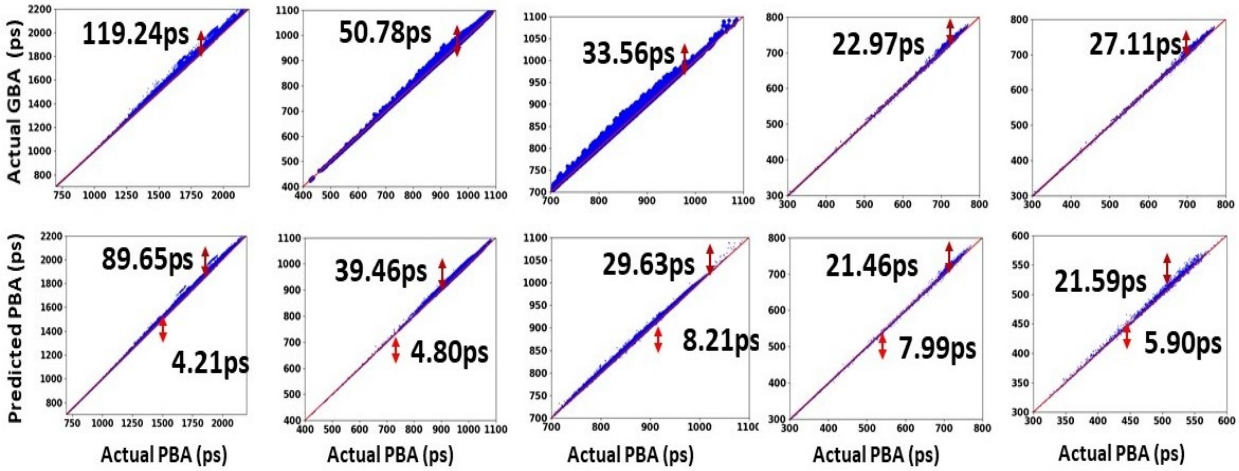


Fig. 13. Results of Experiment 2 for actual GBA path arrival time (top row) and predicted PBA path arrival time (bottom row) versus actual PBA path arrival time for, in left-to-right order, megaboom, leon3mp, netcard, dec_viterbi and jpeg_encoder.

REFERENCES

[1] A. B. Kahng, M. Luo and S. Nath, "SI for Free: Machine Learning of Interconnect Coupling Delay and Transition Effects", *Proc. SLIP*, 2015, pp. 1-8.

[2] A. B. Kahng, S. Kang, H. Lee, S. Nath and J. Wadhwani, "Learning-Based Approximation of Interconnect Delay and Slew in Signoff Timing Tools", *Proc. SLIP*, 2013, pp. 1-8.

[3] S. S. Han, A. B. Kahng, S. Nath and A. Vydyanathan, "A Deep Learning Methodology to Proliferate Golden Signoff Timing", *Proc. DATE*, 2014, pp. 1-6.

[4] R. Molina, EDA Vendors Should Improve the Runtime Performance of Path-Based Timing Analysis, http://www.electronicdesign.com/eda/eda-vendors-should-improve-runtime-performance-path-based-analysis, May 2013.

[5] A. B. Kahng, "Machine Learning Applications in Physical Design: Recent Results and Directions", *Proc. ISPD*, 2018, pp. 68-73.

[6] T.-W. Huang and M. D. F. Wong, "Timing Closure: Speeding Up Incremental Path-Based Timing Analysis with MapReduce", *Proc. SLIP*, 2015, pp. 1-6.

[7] J. Bhasker and R. Chadha, *Static Timing Analysis for Nanometer Designs: A Practical Approach*, Springer, 2009.

[8] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and Regression Trees*, Chapman & Hall/CRC, 1984.

[9] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, 2009.

[10] J. H. Friedman, "Multivariate Adaptive Regression Splines", *Annals of Statistics* 19(1) (1991), pp. 1-67.

[11] L. Breiman, "Random Forests", *Machine Learning* 45 (2001), pp. 5-32.

[12] *Synopsys, Inc.*, https://www.synopsys.com

[13] *Cadence Design Systems, Inc.*, https://www.cadence.com

[14] *scikit-learn*, http://scikit-learn.org

[15] *PyTorch*, https://pytorch.org

[16] *OpenCores*, https://opencores.org

[17] *RISC-V*, https://riscv.org

[18] TAU Workshop, https://www.tauworkshop.com

[19] M. M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke and C. Zhuo, "The ISPD-2012 Discrete Cell Sizing Contest and Benchmark Suite", *Proc. ISPD*, 2012, pp. 161-164.

[20] *Synopsys PrimeTime User Guide*, http://www.synopsys.com/Tools/Implementation/SignOff/Pages/PrimeTime.aspx

[21] *Cadence Tempus User Guide*. https://www.cadence.com/content/cadence-www/global/en_US/home/tools/digital-design-and-signoff/silicon-signoff/tempus-timing-signoff-solution.html

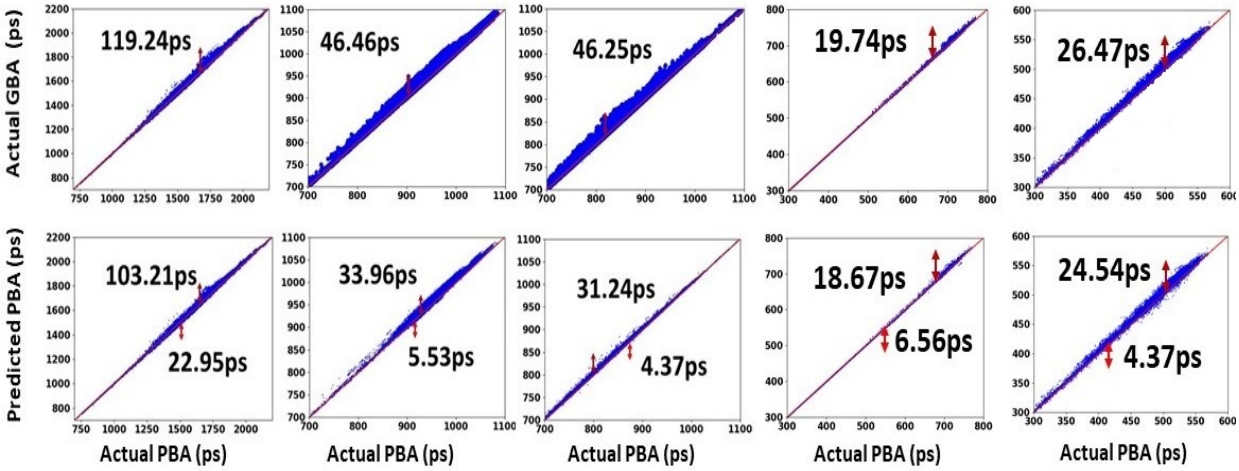


Fig. 14. Results of Experiment 3 for actual GBA path arrival time (top row) and predicted PBA path arrival time (bottom row) versus actual PBA path arrival time for, in left-to-right order, megaboom, leon3mp, netcard, dec_viterbi and jpeg_encoder.

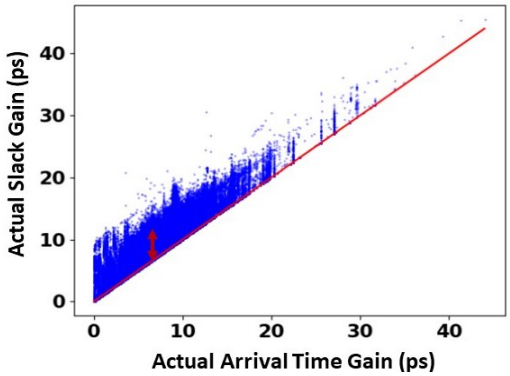


Fig. 15. Arrival time gain versus slack gain for the leon3mp testcase with an average "budget" of 4.46ps (indicated by red arrow) and a maximum "budget" of 17.88ps over 100K endpoints.

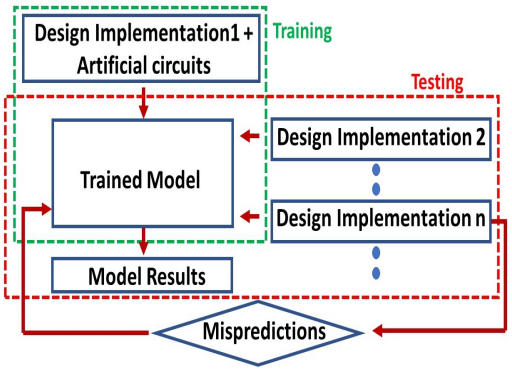


Fig. 16. A potential incremental modeling flow to reduce mispredictions.