Learning-Based Prediction of Embedded Memory Timing Failures During Initial Floorplan Design

Wei-Ting J. Chan[1], Kun Young Chung[2], Andrew B. Kahng[1,3], Nancy D. MacDonald[4] and Siddhartha Nath[3]
CSE[3] and ECE[1] Departments, UC San Diego, La Jolla, CA, USA
Samsung Electronics Co., Ltd.[2], Hwaseong-si, Gyeonggi-do, Korea
ClariPhy Communications[4], Irvine, CA, USA
{wechan, abk, sinath}@ucsd.edu, kun.chung@samsung.com, nancy.macdonald@clariphy.com

Abstract*—Embedded memories are critical to success or failure of complex system-on-chip (SoC) products. They can be significant yield detractors as a consequence of occupying substantial die area, creating placement and routing blockages, and having stringent Vccmin and power integrity requirements. Achieving timing-correctness for embedded memories in advanced nodes is costly (e.g., closing the design at multiple (logic-memory) cross-corners). Further, *multiphysics* (e.g., crosstalk, IR, etc.) signoff analyses make early understanding and prediction of timing (-correctness) even more difficult. With long tool and design closure subflow runtimes, design teams need improved prediction of embedded memory timing failures, as early as possible in the implementation flow. In this work, we propose a learning-based methodology to perform early prediction of timing failure risk given *only* the netlist, timing constraints, and floorplan context (wherein the memories have been placed). Our contributions include (i) identification of relevant netlist and floorplan parameters, (ii) the avoidance of long P&R tool runtimes (up to a week or even more) with early prediction, and (iii) a new implementation of Boosting with Support Vector Machine regression with focus on negative-slack outcomes through weighting in the model construction. We validate accuracy of our prediction models across a range of "multiphysics" analysis regimes, and with multiple designs and floorplans in 28FDSOI foundry technology. Our work can be used to identify which memories are "at risk", guide floorplan changes to reduce predicted "risk", and help refine underlying SoC implementation methodologies. Experimental results in 28nm FDSOI technology show that we can predict P&R slack with multiphysics analysis to within 253ps (average error less than 10ps) using only post-synthesis netlist, constraints and floorplan information. Our predictions are 40% more accurate than the predictions (worst-case error of 358ps and average error of 42ps) of a nonlinear Support Vector Machine model that uses only post-synthesis netlist information.

I. Introduction

Timing closure in modern systems-on-chip (SoCs) is complex and time-consuming, due to multiple iterations between various analyses and design fixes. Early, accurate prediction of post-layout slack can potentially deliver dramatic design turnaround time and design cost reductions. However, to the best of our knowledge, no existing tool can predict slack at an early design stage (in particular, the post-synthesis, physical floorplanning stage).[1] Predicting post-layout slack without physical synthesis or trial placement information is challenging because wire delay must be estimated without spatial embedding information. The prediction problem becomes even more difficult because of (i) embedded memories, and (ii) "multiphysics" analysis.

Embedded memories (SRAMs) complicate SoC physical implementation on several fronts [8] [31]. They occupy significant die area [13] and are typically placed in arrays, which not only makes floorplanning difficult, but also creates placement and routing blockages. Timing analysis and closure are costly, e.g., with respect to cross-corners in low-power, split-rail designs. Hence, despite long tool runs and complex design subflows, SoCs with multiple SRAMs can have unpredictable timing at the post-P&R stage, not to mention in silicon.

Verification of timing correctness in advanced nodes increasingly demands analyses that close the loop across crosstalk, IR and temperature [19] [25], i.e., more than one "physics". We use *multiphysics analysis* to mean performing multiple analyses such as IR, thermal, reliability, crosstalk, etc., and then performing static timing analysis (STA) using reports from these analyses. Timing assessments can vary widely with the specific analyses performed, e.g., turning SI mode on can worsen slack by 100ps due to crosstalk [16]. Figure 1(a) shows slack values of five memories in a small block[2], according to four different analyses that combine IR analysis and STA: (i) STA with no IR analysis; (ii) STA with static IR analysis; (iii) STA with dynamic voltage drop (DVD) IR analysis; and (iv) four iterations of STA with DVD IR analysis, i.e., STA is performed with back-annotated instance-specific DVD IR drop, going around this loop four times. Figure 1(b) shows that across different implementations of the same netlist (i.e., when clock period and maximum transition time constraints are varied), the slack difference between (i) STA with no IR, and (ii) two iterations of STA with DVD IR, can vary by $\sim$15ps depending on the implementations. By closing multiphysics analysis loops, design teams achieve more accurate timing results, but the results of such analyses are non-trivial to predict in early stages of implementation.

We show two examples to illustrate the challenges of predicting post-layout slack. **(1) Sensitivity of slack to spacing between memories.** Figure 2(a) shows a floorplan with five embedded SRAMs, blockages, and a rectilinear standard-cell placement region. Figure 2(b) shows variation of worst timing slack (at any timing endpoint in a given SRAM) across these five SRAMs when the spacing (i.e., channel width) between memories is varied in steps of 10μm. Due to congestion, buffer placement, etc., the difference in slack can be larger than 300ps at a spacing of 10μm, and slacks vary in a highly non-obvious and/or noisy manner as the spacing is changed. **(2) Sensitivity of IR drop map to power pad locations.** Figures 3(a)–(c) show three IR drop maps when the locations of power pads are varied. When power pads are placed uniformly on all edges of the die as in Figure 3(a), the IR map has very few hotspots. When the power pads are placed only on the left and right edges of the die as in Figure 3(b), or on the bottom and top edges as in Figure 3(c), the IR drop map has multiple hotspots. The IR drop map, and timing slacks, have similarly challenging sensitivities to SRAM placement relative to power distribution network stripes (PDN stripes), the availability of buffer placement locations within or near memory channels, etc.

In this work, we apply machine learning to achieve accurate predictive modeling of slacks at embedded SRAM timing endpoints. Given *only* a post-synthesis netlist, constraints and a floorplan, we predict (i) post-P&R slack, and (ii) slack with multiphysics analysis, of SRAMs.

Figure 4 shows the stages of physical implementation that we must comprehend with our modeling, as well as the stages from which we can extract available modeling parameters. We estimate the combined effects of placement, clock network synthesis, routing, extraction and timing using our modeling function f as shown in the figure. Our

[1] A long history of RTL signoff and RTL planning tools is best exemplified today by Atrenta SpyGlass [24], which performs early analysis of designs but uses its own simplified placement, clock tree synthesis and routing engines that do not necessarily match production back-end tool outcomes.

[2] We place only one power/ground pad pair at the south edge for this testcase (OpenCore *THEIA*) to emulate a severe voltage-drop situation. The signoff clock period for this example is 3.5ns.

978-1-4673-9569-4/16/$31.00 ©2016 IEEE

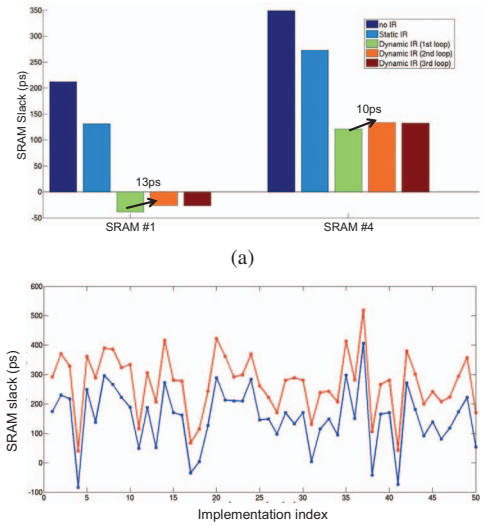


Fig. 1. Multiphysics analysis. (a) SRAM slack with (i) no IR, (ii) static IR, (iii) dynamic voltage drop (DVD) IR, and (iv) four iterations of DVD IR and STA. (b) Difference in slack between (i) no IR and (ii) STA + DVD IR with two loops. The indices in the x-axis of (b) refer to different implementations when clock period and maximum transition time constraints are varied.

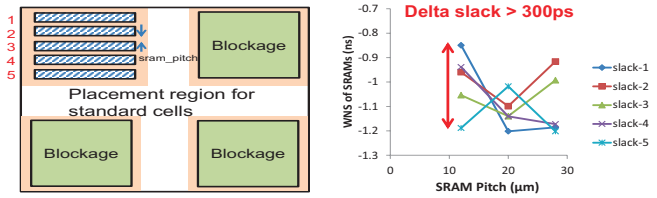


Fig. 2. Sensitivity of slack to spacing between SRAMs: (a) floorplan, and (b) slack variation.

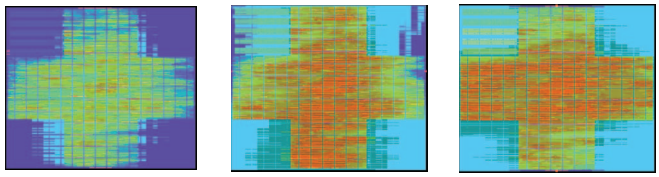


Fig. 3. Sensitivity of IR drop map to power pad placement: (a) on all four edges of layout; (b) left and right edges only; and (c) bottom and top edges.

work envisages two basic use scenarios. (1) For products in the early planning stage, our predicted slacks enable floorplans and constraints – as well as physical implementation methodology – to be adjusted to prevent post-layout timing failures on SRAMs. (2) For products in the production stage, our model enables designers to filter out floorplans and constraints that have high risk of post-layout timing failures under voltage and frequency scaling, or process variation. Our model can prevent costly iterations of floorplan and constraint adjustments.

Our main contributions are summarized as follows.

- To the best of our knowledge, we are the first to propose a modeling methodology that can effectively predict post-P&R slack values at endpoints on embedded SRAMs, using information available at the floorplanning stage. Our model applies machine learning techniques to predict post-P&R slack within a worst-case error of 224ps and average error of 4.0ps across all designs and floorplans tested in a 28nm foundry FDSOI technology.

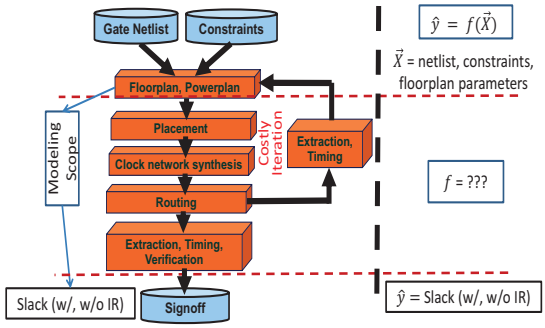


Fig. 4. Traditional IC design flow from [14], with dotted horizontal lines bounding the scope of our model. We comprehend multiple stages of the physical design flow in our model, which is represented by the function f in the figure.

- We confirm the robustness of our prediction methodology by predicting slack values *after* multiphysics analysis – a very difficult prediction problem – to within a worst-case error of 253ps and average error of 9.0ps. A model that uses information from the post-synthesis netlist results in multiphysics slack worst-case prediction error of 358ps.[3]
- We automate using commercial EDA tools the extraction of relevant model parameters, and prediction of timing failure risks, from given netlist, constraints and floorplan context. By predicting multiphysics slack for every embedded memory endpoint, our model enables early filtering and improvement of floorplans that would otherwise eventually lead to timing failures at the post-layout and signoff stages.
- We advance application of machine learning for predictive IC design with a new implementation of the Boosting technique that uses Support Vector Machines (SVMs) as *weak learners*. We also propose a weighting strategy for negative-slack outcomes during our model construction, to accurately focus our model on avoidance of critical timing failures. SVM in Boosting reduces worst-case prediction errors by 30ps relative to use of SVM only.

In the following, Section II reviews related literature, while Section III describes our multiphysics analysis methodology, selection of modeling parameters and our modeling methodology with machine learning techniques. Section IV describes our testcases, design of experiments and results. We give conclude in Section V.

II. Related Work

Very few previous works predict timing of SRAMs in early stages of the design flow. We categorize previous works into (i) P&R timing prediction from netlist, (ii) applications of machine learning for prediction of physical design metrics, and (iii) applications of Boosting [7] and Support Vector Machine (SVM) [9] for regression.[4]

In the category of **post-P&R timing prediction from netlist**, Alpert et al. [2] propose the adoption of physical synthesis in design flows to have better correlation with post-layout metrics such as worst negative slack (WNS), at the post-synthesis stage. Alpert et al. [3]

[3]We use parameters $N1$ through $N6$ from Table I and use three different modeling techniques – LASSO, linear SVM and SVM with a Radial Basis Function (RBF) as kernel – to predict multiphysics slack. The worst-case (resp. average) prediction errors are 565ps (resp. 87ps) for LASSO, 412ps (resp. 55ps) for linear SVM method, and 358ps (resp. 42ps) for SVM with RBF kernel. The SVM model with RBF kernel has smaller prediction errors as compared to those of linear SVM and LASSO models. Therefore, we compare our results with those from the SVM model with RBF kernel.

[4]As noted in the Conclusions below, an eventual target for methods such as what we describe here is the prediction of embedded memory defectivity in silicon (and, floorplan guidance to correct or avoid such defectivity). The difficulty and significance of such a prediction is noted in such works as [31]. While we have not had access to SoC design and product/test engineering data that would allow us to attempt this future application, our work has been motivated by (and kept consistent with) such an end goal.

propose analytical buffered delay models in the presence of blockages and report errors within 1% for three-pin nets. The authors propose integration of their models within a floorplanner for fast and accurate estimation of timing. Clarke et al. [4] propose detection of congestion-induced timing issues during synthesis, and prevention of congestion by avoiding decomposition of complex cells such as MUX and XOR to NAND and AOI cells during logic synthesis so as to reduce pin counts. Hutton and Karchmer [11] propose efficient design-space exploration of metrics such as operating speed, power and area during synthesis of FPGAs. The authors propose to create models using regression on existing synthesized designs. Jones et al. [12] derive wireload models to estimate wire delay due to parasitics at the placement stage. The authors propose to divide the block into equal-sized regions, perform Steiner tree routing and use Rent's rule for fanout distributions in each region. Kim et al. [17] characterize standard cells and parasitics at different temperatures and propose thermal-aware delay models at the floorplanning stage. Vujkovic [22] proposes the use of multiple wireload models during synthesis for better correlation of post-layout wire delay. Yaldiz et al. [23] develop a closed-form model of SRAM latency that comprehends inter- and intra-die process variations. The authors demonstrate accuracy of their models in 90nm and report errors within 15%.

In the category of **applications of machine learning to predict physical design metrics**, Huang et al. [10] use SVM classification to predict defects in analog chips at the post-silicon stage. They use low noise amplifiers as their test circuit and are able to predict defects within an error of 2.9%. Kahng et al. [15] propose use of nonlinear machine learning-based models to estimate cell delay and slew under noise in the power delivery network (PDN), and clock skew after clock tree synthesis.

In the category of **applications of Boosting and SVM for regression**, Kotisantis et al. [18] create an ensemble of regressors by using Bagging [9], Boosting with SVM without a kernel, and Random Forests [9]. The authors then combine outcomes of each regressor using weights that are proportional to the inverse of the error of the outcomes from each regressor. In their method the weights are calculated based on error in the test set, and not on the training set. Ogutu et al. [20] compare Random Forests, Boosting with a linear regressor, and SVM to predict genomic breeding values. The authors conclude that SVM is more accurate than the other two techniques. Our implementation of Boosting builds on [7] [18] by using SVM with a nonlinear kernel.

III. Our Methodology

We now describe the key elements of our work: multiphysics analysis flow, model parameter selection, and machine learning-based modeling methodology. We also note how our analyses and modeling flows would be reproduced in a new environment.

A. Multiphysics STA

Figure 5 shows our multiphysics analysis flow. Due to the very large number of testcase implementations, we focus on an IR-STA multiphysics analysis loop.[5] We perform STA using Synopsys *PrimeTime-SI* (PTSI) [32]. The inputs to the tool are Liberty timing libraries characterized at multiple voltage corners, Verilog netlist of the design, SPEF parasitics [30] with coupling capacitances, and Synopsys Design Constraints (SDC) [5] with timing constraints as well as back-annotated rail voltages of all instances based on the IR drop map. Note that STA is always performed with SI enabled in our flows.

IR (voltage drop) analysis is the first dimension of multiphysics analysis that must be joined with timing analysis. To assess the vulnerability of various floorplans to timing failures from IR drops, we parameterize the stripe width and pitch of on-chip PDNs, the width of power rings, the metal layers that are used for the PDN stripes, and the placement of memories relative to the nearest power pad. To supply power to memories, we generate secondary meshes on a metal layer that is different from the ones used for PDN stripes. For standard cells, we use M2 metal layers to connect PDN stripes. We connect power meshes and stripes in the lower metal layers to the upper metal layers (M9 and M10) through via stacks, as in normal SoC methodology. We perform vectorless dynamic voltage drop (DVD) analysis using ANSYS *RedHawk* [28]; inputs consist of the post-layout design database, technology Layout Exchange Format (LEF) [29] files, Liberty timing libraries, and the minimum and maximum, rise and fall arrival timing windows of all signal pins as reported from *PrimeTime-SI* [32]. Our DVD IR analysis is vectorless due to lack of representative simulation vectors; to our understanding, this reflects common industry practice. We place power pads uniformly along the block periphery so that the IR drop tends to be worst at the block center.

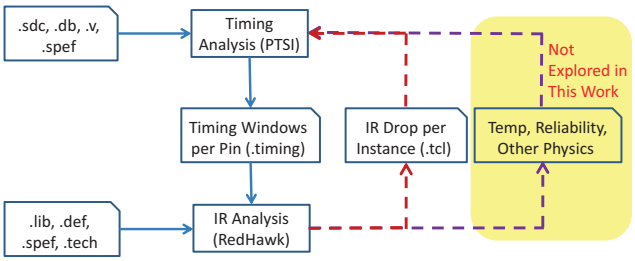


Fig. 5. Our multiphysics analysis flow.

Once we have obtained an IR drop map, we back-annotate individual cell instances with rail voltage in *PrimeTime-SI*, and perform STA using timing libraries that have been characterized at multiple voltage and temperature corners using Synopsys *SiliconSmart* [32]. Again, we view this STA as "multiphysics" in its integration of the IR drop map on a per-instance basis. The standard practice in industry is to perform the above-described multiphysics analysis once. But, recall from Figure 1(a) above that more than one iteration can help remove pessimism in timing analysis by up to 15ps.

B. Model Parameter Selection

We use model parameters that span netlist structure, floorplan context and layout constraints. The modeling problem is high-dimensional when we consider multiple knobs in commercial tools, as well as multiple netlist and layout context parameters. To make our modeling methodology practically applicable, we focus on only those parameters that we have so far found to affect modeling accuracy. We assess the sensitivity of slack to each parameter independently by varying values of one parameter at a time and keeping the remaining parameters the same. We also assess the combined impact of various parameters on the slack of memories using *variance inflation factor* (VIF) [1]. We choose parameters whose VIF values are less than 0.5 [15] and let the modeling techniques (described in Section III-C) combine relevant parameters. Some of our parameters are for the entire layout, whereas the remaining parameters are for each memory instance so that the modeling can capture variable number of memories in the netlist, variations of floorplans, and the placement of memories within these floorplans. Table I lists our modeling parameters. The first column gives the parameter identifier; the second column describes the parameter; the third column shows whether the parameter is of type netlist, floorplan or constraint; and the last column indicates that the parameter is obtained per memory instance when it is "Yes". Some of our modeling parameters are based on guidance provided in [2], [4] and [15].

[5]This can be extended to more complete multiphysics analyses that include temperature and reliability - e.g., using ANSYS Sentinel-TI and RedHawk-SEM. While we have prototyped such analyses, they are cumbersome with available tools, and we do not report any studies here.

TABLE I
PARAMETERS USED IN OUR MODELING.

Parameter	Description	Type	Per-memory?
N1	Max delay across all timing paths at the post-synthesis stage	Netlist	Yes
N2	Area of cells in the intersection of startpoint fanout and endpoint fanin cones of max-delay incident path	Netlist	Yes
N3	Number of stages in the max-delay incident path	Netlist	Yes
N4, N5, N6	Max, min and average product of #transitive fanin and #transitive fanout endpoints	Netlist	Yes
N7	Width and height of memory	Netlist	Yes
FP1	Aspect ratio of floorplan	Floorplan	No
FP2	Standard cell utilization	Floorplan	No
FP3, FP4	PDN stripe width and pitch	Floorplan	No
FP5	Size of buffer screen around memories	Floorplan	No
FP6	Area of blockage (%) relative to floorplan area	Floorplan	No
FP7, FP8	Lower-left placement coordinates of memories	Floorplan	Yes
FP9, FP10	Width, height of channels for memories	Floorplan	Yes
FP11	#memory pins per channel	Floorplan	Yes
C1	Sum of width and spacing of top-three routing layers after applying non-default rules (NDRs)	Constraint	No
C2	% cells that are LVT	Constraint	No
C3, C4	Max fanout of any instance in data and clock paths	Constraint	No
C5, C6	Max transition time of any instance in data and clock paths	Constraint	No
C7	Delay of the largest buffer expressed as FO4 delay	Constraint	No
C8	Clock period used for P&R expressed as FO4 delay	Constraint	No
C9	Ratio of clock periods used during synthesis and P&R	Constraint	No

C. Modeling Techniques

Recall from Figure 4 that we seek to model (i) multiple stages of the physical design flow such as placement, clock network synthesis, routing, extraction, etc., (ii) inherent noise in commercial tools, and (iii) a very high-dimensional space of parameters that span across netlists, floorplan contexts and timing constraints. Interactions between parameters are complex, e.g., an increase in PDN stripe density can cause a large congestion on upper metal layers and thereby increase coupling capacitances which will ultimately result in timing failures even when the IR drop is small. The type of timing analysis can contribute to large difference in slack, e.g., turning SI mode on can worsen slack by 100ps or more [16].

We use both linear as well as nonlinear machine learning techniques. We use LASSO regression with L1 regularization [21] as a linear technique, and Support Vector Machine (SVM) regression [9] with a Radial Basis Function (RBF) kernel [9], Artificial Neural Networks (ANN) [9], and Boosting [7] with a weak SVM learner as the nonlinear techniques. The Boosting learning technique combines predictions of multiple weak learning techniques to create an accurate learning model. Learning techniques such as linear classification and regression trees are used commonly as weak learners. For a comprehensive discussion on LASSO, SVM, ANN and Boosting, see [9]. For each technique, we use training and validation data sets that are 50% and 10% of the total data points, respectively, and we search for values of hyperparameters using grid search such that the training and validation mean-square errors (MSEs) are comparable. For SVM with RBF kernel, the hyperparameters are ϵ along with the cost C that control the margin errors of the support vectors, and the width parameter γ of the RBF kernel. For ANN, we define the architecture as one input and one output layer and two hidden layers. The hyperparameters are the number of epochs for back propagation and the number of neurons per hidden layer. For LASSO, the hyperparameter is the regularization coefficient λ. For Boosting with SVM, the hyperparameters are ϵ, C, γ, and the number of iterations.

For each machine learning technique, we perform five-fold cross-validation so as to make the models generalizable. We normalize the parameters to within $[0, 1]$ before we proceed with modeling.[6] The nonlinear techniques (SVM, ANN and Boosting) help to capture complex interactions between parameters. Our preliminary studies indicate that SVM with a RBF kernel method achieves higher accuracy (less than 300ps worst-case error) than linear SVM without a kernel (more than 335ps worst-case error). Therefore, we use SVM with a kernel method. The LASSO technique has large modeling error (greater than or equal to 300ps) when the number of parameters is larger than five. So, we use the linear LASSO technique to make predictions with a simplified model and use the outcomes of this linear technique as the bias in the final step in which we combine outcomes of all the techniques using Hybrid Surrogate Modeling (HSM) [15] to obtain the final predicted slack of each memory instance. Even though the procedure to combine predictions from various linear and nonlinear techniques is not obvious, the HSM technique enables us to combine the predictions using appropriate weights and improve overall prediction accuracy. Figure 6 shows our modeling flow. We implement our modeling in *Matlab vR2013a* [26] using default toolboxes for ANN and LASSO, the open-source *libsvm* [6] toolbox for SVM, and our own implementation of Boosting.

Figure 7 shows our high-level implementation of Boosting.[7] We implement Boosting with weak SVM learners as follows.

- Initially, we set the weight W_0 of all training datapoints to be uniform, i.e., $W_0 = 1/N_{tr}$, where N_{tr} is the number of training data points.
- We use SVM as a weak learner by restricting the grid search to only three different values of each hyperparameter.
- We calculate the error e_i for the i^{th} stage using the validation set and W_i values of datapoints are set for the subsequent $(i+1)^{st}$ stage as $exp(0.5 \log(\frac{1-e_i}{e_i}))$ when the error in slack prediction is greater than or equal to 50% of the clock period, and are set to 1 otherwise.
- To make our predictions pessimistic on datapoints for which the actual slack is negative, we increase W_i by a factor of five when the predicted slack value for such a datapoint is positive.
- We terminate when worst-case error in the validation set is less than 20% of the clock period or when the number of iterations reaches $k = 40$.[8]
- We combine outcomes of each iteration using coefficients β_i (determined by using least-squares regression), where $i = 1, ..., k$, to determine the final outcome of Boosting.

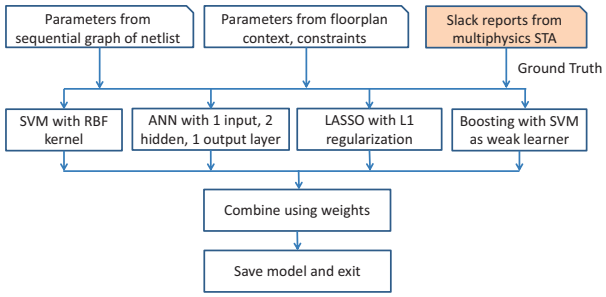


Fig. 6. Modeling flow with linear and nonlinear regression techniques.

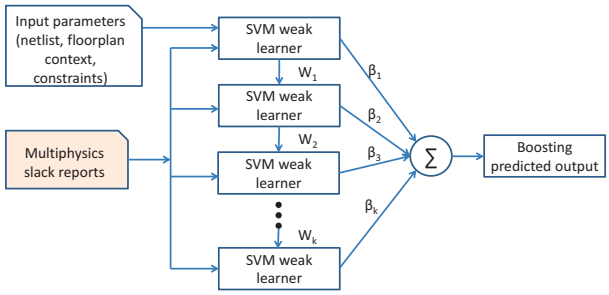


Fig. 7. Flow with Boosting [7] with weak SVM learners.

[6]We have tried normalization using z-scores to within $[-1, 1]$ as well. The predicted values change by less than 0.5%. Therefore, we use normalization to within $[0, 1]$ in our experiments.

[7]Of possibly independent interest is that to our knowledge, Boosting with SVM as a weak learner (i.e., regressor) has not been tried before in the machine learning and VLSI CAD literatures. Our scripts are available at http://vlsicad.ucsd.edu/Riskmap/.

[8]With values of $k > 40$ we do not observe significant improvement in error.

From the above discussion, the reader will note that the proposed methodology will in practice use post-P&R databases of various projects taped-out in a given (technology node, library, tool flow) as the basis of netlist structural analyses, floorplan structural analyses, and multiphysics performance analyses. Designers of new projects in the same technology node would use our model to filter out floorplans and constraints that can cause timing failures. In a new technology or design environment, the one-time initial model fitting effort that we describe above must be performed.

IV. Validation and Results

In this section, we describe our testcases, design of experiments and present our modeling results.

A. Testcases

We have developed a generator to create testcases to vary (i) the number of SRAMs in the netlist,[9] (ii) the floorplan context such as aspect ratio, utilization, buffer screens, PDN structure, etc., and (iii) the placement of SRAMs in the floorplan. Figure 8 illustrates various parameterizations of floorplans in our testcase generator, using the floorplans shown in Figures 10(a) and 12(a). We can independently change the width and height of buffer screens around SRAMs or blockages, the dimensions of each blockage, and the area for standard cell placement.

Our netlists contain both logic and SRAMs. For logic, we use open-source designs such as *THEIA*[10] and *nova* from OpenCores [27], our own artificial testcases with an embedded processor, and blocks from OpenCores such as *aes_cipher_top* and *reed_solomon_codec*. We perform synthesis using 28nm foundry FDSOI libraries and Synopsys *Design Compiler vI-2013.12-SP3*.[11] Table II summarizes our netlists with post-synthesis metrics.

TABLE II
Description of our netlists.

Netlist	Clock Period (ns)	#Std Cells	#SRAMs	Logic Area (μm^2)	SRAM Area (μm^2)
THEIA_v0	3.0	147274	40	157416	347252
THEIA_v1	2.7	146505	5	157068	40027
THEIA_v2	3.0	146914	6	157012	48032
THEIA_v3	3.0	146243	8	156212	64043
THEIA_v4	3.0	146606	10	155991	80054
nova	2.0	66031	5	68970	25117
artificial	2.0	201015	6	213075	14925

Figure 9 illustrates a PDN structure used in our testcases. We use metal layers M9 and M10 for the power ring around the core; we also use M9 and M10 for the top-level power mesh. We use M6 to generate secondary meshes to supply power to SRAMs, and M2 to connect standard cells to the VDD and ground rails. From the post-P&R databases we generate the routed Design Exchange Format (DEF) [29] file using Synopsys *IC Compiler vH-2013.03-SP3* and provide it as an input to ANSYS/Apache *RedHawk v10.1.7* along with technology LEF and Liberty timing libraries. We use Synopsys *PrimeTime-SI vH-2013.06-SP2* to obtain timing windows of all signal pins.

B. Design of Experiments

Using our generator described in Section IV-A and netlists in Table II, we create various testcases in which we vary floorplans, PDN structures and constraints. Table III lists the parameters we vary in our design of experiments. We vary the standard-cell placement

[9]We use single-port SRAMs of two different sizes from the 28nm FDSOI foundry libraries.

[10]We have used the original OpenCores *THEIA* design as well as modified versions of the design. In the modified designs (*THEIA_v1*, ..., *THEIA_v4*), we vary the number of SRAMs. The unmodified design is *THEIA_v0*.

[11]Modeling based on physical synthesis tools such as *Design Compiler Topological* and *RTL Compiler Physical* is part of our ongoing work.

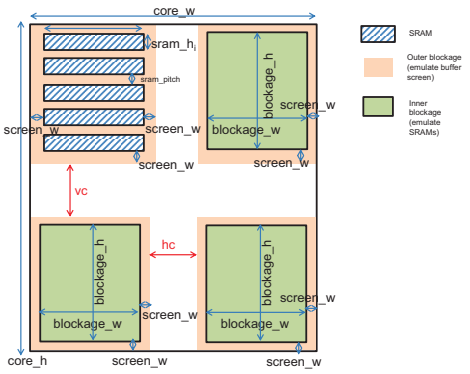


Fig. 8. Parameterized floorplan used to generate testcase instances.

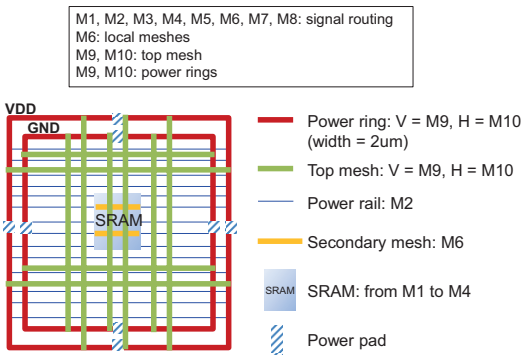
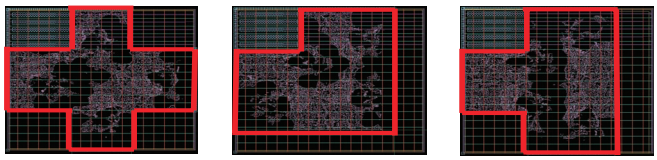


Fig. 9. Example of PDN structure in our testcase with SRAMs.

region to have cross-, L- and T-shapes as shown in Figures 10(a)-(c). Each of these region shapes changes P&R tool outcomes since it changes degree of nonconvexity (i.e., number of nonconvex corners) in the placement region, as well as the placement of IO pins. For example, the cross-shaped floorplan has more nonconvex corners and is expected to have higher congestion near these corners as compared to the L-shaped floorplan.

To emulate real designs from the industry, we frame our experiments in the context of a general, "tic-tac-toe" floorplan. We divide the block with two shiftable gridlines in each axis; each of the nine resulting gridcells can be fully or partially occupied by essential components of a floorplan, that is, hard macros, standard cells or blockages. The tic-tac-toe implementation (i) enables generality and parameterizability, (ii) enables the ability to explore a discrete design space systematically, and (iii) captures how designers tend to floorplan their blocks. Figure 11 shows an example instance of a tic-tac-toe floorplan. (Note that the tic-tac-toe framework allows us to explore floorplans either at the die-level or block-level, but not in between.)



(a) (b) (c)

Fig. 10. Variations in floorplans in our testcases: (a) cross-shape, (b) L-shape, and (c) T-shape. The red lines are used to highlight these shapes.

We create multiple variations of floorplans for netlists with five, six, eight, 10 and 40 memories. Figures 12(a)–(f) show examples of six variations that we generate. All of these floorplans can be created with the tic-tac-toe implementation. Specifically, we create the floorplans with eight, 10 and 40 memories using this implementation. For the tic-tac-toe implementation, we focus on testcases with more than eight memories. Note that our modeling parameters listed in Table I can handle variations of floorplans shown in Figures 10 and 12 because

we include parameters that are floorplan-specific as well as memory instance-specific. We allow only buffer instances to be placed within the buffer screens around SRAMs.

A real industry flow will run *Design Compiler* in topographical mode with floorplan constraints and generate a DEF with placement of standard cells. However, this requires a license which is not available to us. We denoise each P&R run by varying the parameters by $\pm 0.5\%$ from its value. We generate a total of 2515 data points for modeling, out of which we use 1248 (50%) data points for training, 226 (9%) data points for validation, and the remaining 1041 (41%) data points for testing. We challenge our modeling by testing on all data points of testcases *nova* and values of aspect ratio, utilization, and PDN width and height, which are not used for training.

We use the multiphysics analysis flow described in Section III-A, and the modeling methodology described in Section III-C to derive our model. Each P&R and analysis run requires approximately 10 hours using a single core, and the training time is around three hours for 2515 data points on an Intel Xeon E5-2640 2.5GHz when using four cores. The testing time for 1041 data points is less than two minutes.

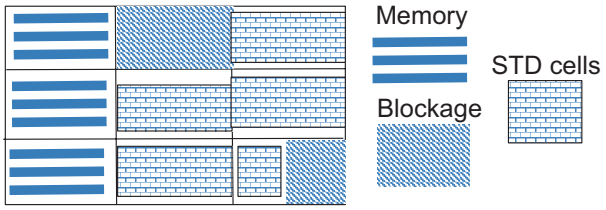


Fig. 11. Example of a floorplan enumerated with tic-tac-toe implementation.

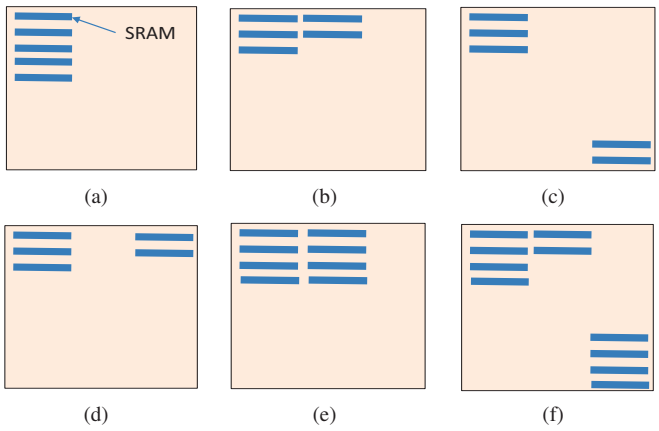


Fig. 12. Examples of memory placements in our testcases: (a) 5×1 vertical stacking, (b) 3×1, 2×1 side-by-side arrays at upper-left, (c) 3×1, 2×1 arrays at upper-left and lower-right, (d) 3×1, 2×1 arrays at upper-left and upper-right, (e) 4×1, 4×1 side-by-side arrays at upper-left, and (f) 4×1, 2×1 side-by-side arrays at upper-left and 4×1 at lower-right.

We conduct three experiments to validate our model. In all of our experiments we use datapoints from testcases *THEIA_v0* and *nova* exclusively for testing, i.e., no datapoints from these two testcases are used to train the model.

- **Experiment 1** tests accuracy of our model in predicting post-P&R slack values of SRAMs. With the training data generated by the design of experiments described above in Table III, we apply our modeling flow described in Figure 6 to predict the post-P&R memory timing slack values. We also compare the accuracy of various modeling techniques and present our results in Table IV.
- **Experiment 2** tests accuracy of our model in predicting slack values with multiphysics analysis. We use the same design of experiments and modeling flow as in Experiment 1 to predict SRAM timing slack values after annotating IR drop from the *RedHawk* reports to cell instances.
- **Experiment 3** tests fidelity of our model in providing floorplan guidance to reduce timing failures at signoff with multiphysics analysis. We report the confusion matrix of timing pass or fail predictions, again using the same design of experiments and modeling flow.

TABLE III
OUR DESIGN OF EXPERIMENTS.

Parameter	**Value(s)** (* is default)
Aspect ratio	{1.2, 1.1, 1.0*, 0.8}
Utilization (std cells)	{40%, 50%*, 60%, 70%}
PDN stripe width	{0.5, 0.75, 1.0*, 1.5, 2.0, 2.5, 3.5}μm
PDN stripe pitch	{7, 15, 20, 30*, 40}μm
SRAM spacing (channel width)	{6, 8, 12, 16, 20*, 24}μm
Buffer screen width	{10, 12, 14*, 16}μm
Routing metal layers	{7, 8*}
Memory placement	{Face-to-face*, face-to-back}
Clock period	*THEIA_{v0, v1, v3, v4}* = {3.0, 3.5*, 4.0}ns *THEIA_v2* = {3.0*}ns *nova* = {3.2*, 3.7, 4.2}ns *artificial* = {2.0*}ns
Max transition	{200*, 240, 280}ps
Max fanout	{8*, 10}
Threshold voltage mixes	{{LVT}, {LVT, RVT}*, {RVT}}
Clock buffer sizes	{{X32}*, {X32, X24}, {X32, X24, X16}}
NDRs on clock nets	{1W1S*, 2W2S, 3W3S, 3W2S, 2W3S}

C. Results

For all of our experiments, we separately report modeling errors (i.e., predicted slack − actual slack) for both training and test datasets. Figures 13(a) and (b) show the ground truth that we predict. Figure 14 compares post-synthesis slack of SRAMs with post-P&R slack. Due to changes in constraints and implementations, post-P&R slack has no apparent correlation with post-synthesis slack.

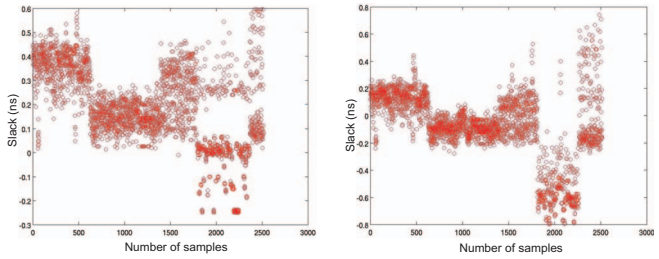


Fig. 13. Ground truth data. (a) Slack at post-P&R stage without multiphysics analysis, and (b) slack with multiphysics analysis.

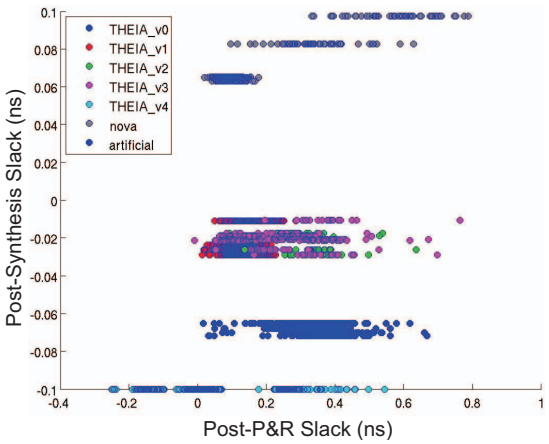


Fig. 14. Slack at post-synthesis stage vs. post-P&R stage across six of our testcases. There is no correlation due to constraints and implementations. (Compare with Figure 15, below.)

Results of Experiment 1. Table IV shows error metrics of our modeling techniques on the test dataset for the various machine learning techniques. Rows 4 and 5 in the table show that the worst-case error in slack prediction reduces by 30ps with our implementation of Boosting with SVM regressors, compared to the SVM-only technique. Figure 15(a) shows predicted versus actual slack values of memories at the post-P&R stage. Our model has worst-case error of 224ps (48%)[12] and average error of 4.0ps (7.2%). Note that most of the predicted slack values (when the actual slack values are negative) are below the solid black line (i.e., line of perfect correlation) as a result of negative-slack weighting strategy. Figure 15(b) shows a histogram of error in slack prediction in the test dataset, and Figure 15(c) shows the outcome of our negative slack weighting strategy during our model construction, i.e., greater magnitudes of the negative slack values have pessimistic predictions.

TABLE IV
ERROR METRICS OF MODELING TECHNIQUES USED IN OUR EXPERIMENTS.

Technique	Min Error (ps)	Max Error (ps)	Mean Error (ps)	Standard Deviation (ps)	Mean-Square Error
LASSO	-380.5	281.6	-86.7	64.1	11.6
ANN	-250.6	272.9	-8.5	60.1	3.7
SVM	-243.7	252.9	-9.0	55.2	3.1
Boosting	-253.7	200.8	-5.1	55.7	3.1
HSM	-223.1	223.7	-4.0	58.9	3.5

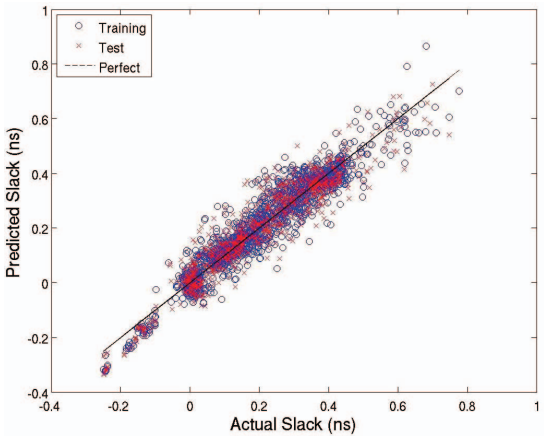
Results of Experiment 2. Figure 16(a) shows predicted versus actual slack values of memories with multiphysics analysis. Our model has worst-case error of 253ps (44%) and average error of 9.0ps (5.9%). Figure 16(b) shows a histogram of error in slack prediction in the test dataset. Even though our worst-case error is large, only a small number of predictions have error greater than 100ps. Some of these predictions are more pessimistic due to our negative-slack weighting strategy. Since prediction of slack with multiphysics analysis is more difficult than predicting post-P&R slack, the errors are larger than those in Experiment 1.

Results of Experiment 3. Figure 17 shows the confusion matrix of our predictions on the test set. Our predictive model of SRAM slack values with multiphysics analysis has few ($\sim$3%) false negatives (fn), that is, pessimistic predictions in which we provide guidance to change a floorplan that is actually not required. Our model also has few false positives (fp), that is, cases for which our model incorrectly deems a floorplan to be good. Such wrong predictions have a $\sim$4% incidence. The number of true positives (tp), that is, both the predicted and actual slack values are positive, is 584. The number of true negatives (tn), that is, both the predicted and actual slack values are negative, is 384.

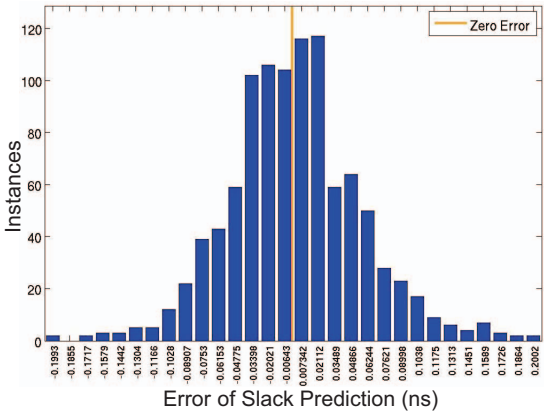
In our model, the *precision* [9] (i.e., the ratio of tp to the sum of tp and fp) is 93.3%, and the *recall* (i.e., the ratio of tp to the sum of tp and fn) is 95.0%. Similarly, the precision for negative slack datapoints (referred to as *negative predictive value* in the machine learning literature) is 92.5% and the recall for negative slack datapoints (referred to as *specificity* in the machine learning literature) is 90.1%. Based on these large values of precision and recall metrics, we believe that our model can provide guidance to designers on the risk of SRAM timing failures with high fidelity.

V. CONCLUSIONS AND FUTURE WORK

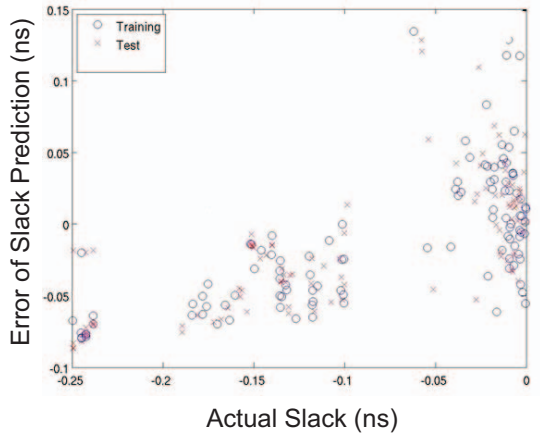
Early prediction of post-layout timing failures is important to reduce design cost and turnaround time. However, this prediction problem is very difficult as it must comprehend tool flows, noise, and the physics used during timing analysis. We propose a machine learning-based methodology to predict post-P&R slack of SRAMs at the floorplanning stage, given only a netlist, constraints and floorplan context. We demonstrate that our methodology can be extended to predict slack with multiphysics (STA and DVD IR) analysis. We develop a new implementation of Boosting with SVM in which we use a negative-slack bias strategy. This strategy guides model predictions to be less optimistic when the actual slack values are negative. We report worst-case modeling error of 253ps in predicting slack with multiphysics analysis, and average error of 9.0ps. The number of predictions with



(a)



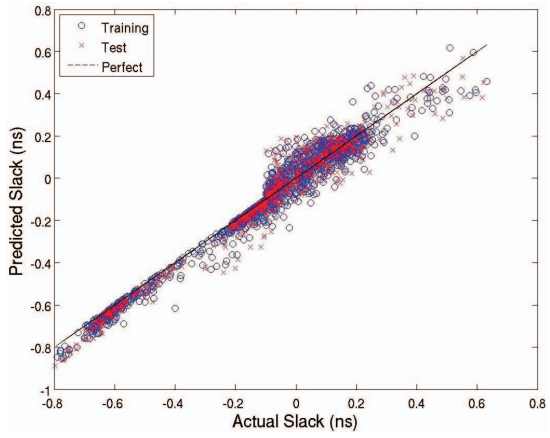
(b)



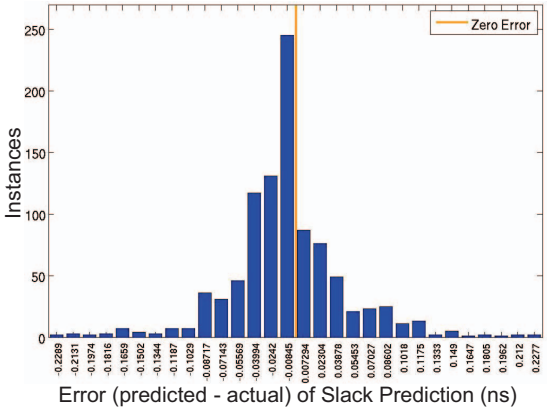
(c)

Fig. 15. Accuracy of our model in predicting post-P&R SRAM slack values with HSM. (a) Scatter plot of actual and predicted data points in training and testing, (b) error distribution in the test dataset, and (c) effect of weighting strategy for negative slack values. Note in (c) that when actual slack values are less than -0.15ns, the error values are negative, i.e., the predicted slack is always pessimistic as compared to the actual slack.

[12]The worst-case error in Experiment 1 occurs when the actual slack is 466ps, whereas the predicted slack is 242ps. The error is $242ps - 466ps = -224ps$; we calculate the magnitude of relative error for this datapoint (relative to the actual slack) as $\frac{224ps}{466ps} = 48\%$. We measure average error as the mean of all absolute errors, and average percentage error as the mean of magnitudes of relative errors (relative to actual slack values) expressed as a percentage.



(a)



(b)

Fig. 16. Accuracy of predicted multiphysics SRAM slack values with HSM. (a) Scatter plot of data points in training and testing, and (b) error distribution for the test dataset.

		Actual	
		Pass	Fail
Predicted	Pass	584	42
	Fail	31	384

Fig. 17. Confusion matrix of our predictions with HSM. False positives (42) are optimistic predictions, while false negatives (31) are pessimistic predictions.

error greater than 100ps are few ($\sim$15) in our test dataset. Fidelity of our predictions is high as measured by the precision and recall metrics. We believe that SoC designers can use our methodology to avoid floorplans and constraints that may cause timing failures at signoff. Our ongoing works include: (i) applying our methodology to product/test engineering data from an SoC design company, (ii) predicting defectivity in silicon and providing floorplan guidance to correct or avoid such defectivity, (iii) performing physical synthesis and use its outcomes for modeling, and (iv) developing a flow to consider SRAM variability.

Acknowledgments

We thank P. Agrawal of ANSYS and J.-A. Desroses of STMicroelectronics for their generous help with setup and enablement of the iterative DVD analysis and signoff timing flow. This work is supported by Samsung Electronics.

References

[1] P. D. Allison, *Multiple Regression: A Primer*, Thousand Oaks, Pine Forge Press, 1999.
[2] C. J. Alpert, C. Chu and P. G. Villarubia, "The Coming of Age of Physical Synthesis", *Proc. ICCAD*, 2007, pp. 246-249.
[3] C. J. Alpert, J. Hu, S. S. Sapatnekar and C. N. Sze, "Accurate Estimation of Global Buffer Delay Within a Floorplan", *IEEE Trans. on CAD* 25(6) (2006), pp. 1140-1146.
[4] M. Clarke, D. Hammerschlag, M. Rardon and A. Sood, "Eliminating Routing Congestion Issues with Logic Synthesis", *Whitepaper*, Cadence Design Systems, 2011. http://www.cadence.com/rl/resources/white_papers/routing_congestion_wp.pdf
[5] J. Bhasker and R. Chadha, *Static Timing Analysis for Nanometer Designs: A Practical Approach*, New York, Springer, 2009.
[6] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library of Support Vector Machines", *Trans. on Intelligent Systems and Technology* 2(3) (2011), pp. 27:1-27:27.
[7] Y. Freund and R. E. Schapire, "A Short Introduction to Boosting", *J. Japanese Society for Artificial Intelligence* 14(5) (1999), pp. 771-780.
[8] S. Hamdioui, "Testing Embedded Memories: A Survey", *Mathematical and Engineering Methods in Computer Science* 7721 (2013), pp. 32-42.
[9] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, New York, Springer, 2009.
[10] K. Huang, H.-G. Stratigopoulos and S. Mir, "Fault Diagnosis of Analog Circuits Based on Machine Learning", *Proc. DATE*, 2010, pp. 1761-1766.
[11] M. D. Hutton and D. Karchmer, "Early Timing Estimation of Timing Statistical Properties of Placement", *U.S. Patent* No. 8,112,728, 2012.
[12] T. R. Jones, S. L. Crain and J. J. Burkis, "Method for Determining Timing Delays Associated with Placement and Routing of an Integrated Circuit", *U.S. Patent* No. 5,629,860, 1994.
[13] A. B. Kahng, "The Road Ahead: Product Futures", *IEEE Design and Test of Computers* 28(6) (2011), pp. 88-89.
[14] A. B. Kahng, J. Lienig, I. L. Markov and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, New York, Springer, 2011.
[15] A. B. Kahng, B. Lin and S. Nath, "Enhanced Metamodeling Techniques for High-Dimensional IC Design Estimation Problems", *Proc. DATE*, 2013, pp. 1861-1866.
[16] A. B. Kahng, M. Luo and S. Nath, "SI for Free: Machine Learning of Interconnect Coupling Delay and Transition Effects", *Proc. SLIP*, 2015.
[17] M. Kim, B.-G. Ahn, J. Kim, B. Lee and J. Chong, "Thermal Aware Timing Budget for Buffer Insertion in Early Stage of Physical Design", *Proc. ISCAS*, 2012, pp. 357-360.
[18] S. Kotsiantis and D. Kanellopoulos, "Combining Bagging, Boosting and Random Subspace Ensembles for Regression Problems", *Intl. J. of Innovative Computing, Information and Control* 8(6) (2012), pp. 3953-3961.
[19] S. K. Nithin, S. Gowrysankar and S. Chandrasekar, "Dynamic Voltage (IR) Drop Analysis and Design Closure: Issues and Challenges", *Proc. ISQED*, 2010, pp. 611-617.
[20] J. O. Ogutu, H.-P. Piepho and T. Schulz-Streeck, "A Comparison of Random Forests, Boosting and Support Vector Machines for Genomic Selection", *Proc. BioMedical Central* 5(Suppl 3) (2011).
[21] M. Y. Park and T. Hastie, "L_1 Regularization Path Algorithm for Generalized Linear Models", *J. Royal Statistical Society* 69(4) (2007), pp. 659-677.
[22] M. Vujkovic, D. Wadkins, B. Swartz and C. Sechen, "Efficient Timing Closure Without Timing Driven Placement and Routing", *Proc. DAC*, 2004, pp. 268-273.
[23] S. Yaldiz, U. Arslan, X. Li and L. Pileggi, "Efficient Statistical Analysis of Read Timing Failures in SRAM Circuits", *Proc. ISQED*, 2009, pp. 617-621.
[24] Atrenta SpyGlass. http://www.atrenta.com/pg/2/
[25] "Techniques to Accelerate Power and Timing Signoff of Advanced-Node SoCs", *Whitepaper*, Cadence Design Systems, 2014. http://www.cadence.com/rl/Resources/white_papers/power_timing_signoff_wp.pdf
[26] *Matlab*, http://www.mathworks.com
[27] *OpenCores*, http://opencores.org
[28] RedHawk User Guide. https://www.apache-da.com/products/redhawk
[29] LEF DEF Reference Manual. http://www.si2.org/openeda.si2.org/projects/lefdef
[30] Standard Parasitic Exchange Format. https://en.wikipedia.org/wiki/Standard_Parasitic_Exchange_Format
[31] "De-Risking Variation-Aware Custom IC Design with Solido Variation Designer and Synopsys HSPICE", *Whitepaper*, Synopsys Inc., 2010. https://www.synopsys.com/Tools/Implementation/CustomImplementation/CapsuleModule/Solido-HSPICE-wp.pdf
[32] Synopsys EDA Tools. http://www.synopsys.com