Learning-Based Approximation of Interconnect Delay and Slew in Signoff Timing Tools

Andrew B. Kahng[+†], Seokhyeong Kang[†], Hyein Lee[†], Siddhartha Nath[+] and Jyoti Wadhwani[†]
[+]CSE and [†]ECE Departments, University of California at San Diego
{abk, shk046, hyeinlee, sinath, jwadhwan}@ucsd.edu

***Abstract*—Incremental static timing analysis (iSTA) is the backbone of iterative sizing and Vt-swapping heuristics for post-layout timing recovery and leakage power reduction. Performing such analysis through available interfaces of a signoff STA tool brings efficiency and functionality limitations. Thus, an internal iSTA tool must be built that matches the signoff STA tool. A key challenge is the matching of "black-box" modeling of interconnect effects in the signoff tool, so as to match wire slew, wire delay, gate slew and gate delay on each arc of the timing graph. Previous moment-based analytical models for gate and wire slew and delay typically have large errors when compared to values from signoff STA tools. To mitigate the accumulation of these errors and preserve timing correlation, sizing tools must invoke the signoff STA tool frequently, thus incurring large runtime costs. In this work, we pursue a learning-based approach to fit analytical models of wire slew and delay to estimates from a signoff STA tool. These models can improve the accuracy of delay and slew estimations, such that the number of invocations of the signoff STA tool during sizing optimizations is significantly reduced.**

I. Introduction and Motivation

Post-layout timing recovery and leakage power reduction comprise an integral step in low-power physical implementation flows [7] [12] [21]. Performing timing analysis using full static timing analysis (STA) during these optimization loops can pose efficiency limitations. Incremental STA (iSTA) for post-route timing closure and power optimization is a key use context in the physical implementation flow today [11] [13], and is the focus of this paper. A signoff STA tool (ST) will typically support incremental analysis, but invoking the ST repeatedly during a leakage power reduction loop may incur prohibitive runtime costs. Therefore, an internal iSTA tool must be built to match a ST to avoid these limitations.

The internal iSTA tool requires accurate delay and slew so that worst-case timing slacks at endpoints have small errors with respect to corresponding values in the ST. However, our studies show that well-known analytical models for wire delay [2] [3] [16] [23], and wire slew [1] [3] [6] [17] [24] can have large errors with respect to the ST, for two main reasons. First, the ST uses "black-box" modeling of interconnect effects, which makes matching an internal iSTA tool to a ST very challenging. For example, even with the gate slew lookup tables using the actual effective capacitance and input slew values from the ST, the calculated output slew does not match what the ST actually returns in its internal calculations. This was a surprise to R&D engineers at the large semiconductor company that organized the ISPD 2013 gate sizing contest [25]. Second, incorrect slew estimates can lead to large errors due to error propagation along timing paths. Lookup table-based gate slew and delay models can be relatively accurate for individual stages of a timing path. However, as the number of stages increases, and paths contain heterogeneous mixes of cell types and sizes, the errors at individual stages can accumulate, resulting in large positive (overestimation) or negative (underestimation) errors in endpoint slacks when compared with a ST. Figure 1 illustrates the concept of error accumulation along a timing path.

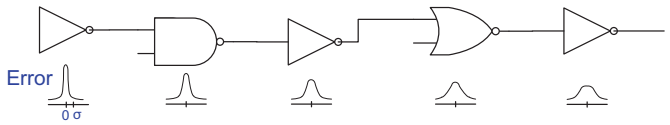

Fig. 1. Propagation of errors along a path.

To minimize divergence of timing results from a ST, internal iSTA tools can invoke the ST periodically to correlate values of gate and wire slew, delay, actual arrival time (AAT), effective capacitance and/or slack, in the process determining correlation "offset" values to minimize timing errors. Therefore, internal iSTA tools must comprehend how to bear the cost of invoking the ST as infrequently as possible.

In this paper, we propose two key techniques, (i) a machine learning-based approach to model wire delay/slew, and (ii) an offset-based timing correlation approach, to obtain an internal iSTA tool which can produce timing values close to those from ST. We explore several analytical models for wire delay/slew and evaluate their accuracies in estimating timing when timing arcs have various sizes and types of cells. We then derive accurate, learning-based models for wire delay and slew for a given ST by using classifiers along with weighted combinations of analytical models. For further compensation of timing differences between an internal iSTA tool and a ST, we explore various offset-based timing correlation approaches [20], which correlate internal timing information (e.g., delay, slew, AAT and slack) with a ST. We evaluate each correlation approach in terms of its inherent tradeoff between runtime and accuracy according to the correlation frequency. Our goal is to minimize the number of ST invocations needed to correlate timing information within an internal iSTA tool.

Our key contributions are as follows.

- We develop machine learning-based models of wire delay/slew for internal iSTA to delay the deviation in endpoint slack from a ST.
- We explore several analytical and learning-based models, and analyze the impact of each model on the timing discrepancy between a ST and an internal iSTA tool.
- We describe an offset-based correlation methodology that improves accuracy in estimating endpoint slack by up to 10× as compared to the prior endpoint slack correlation in [20].

The rest of our paper is organized as follows. In Section II, we describe previous works on incremental timing tools and machine learning techniques used in EDA. Section III describes analytical models used in our work, our methodology to derive learning-based wire delay and slew models, as well as our offset-based correlation methodology. Section IV presents results of our learning-based models and impacts of correlation methodology on timing discrepancy. Section V concludes the paper.

II. Related Works

Academics have developed internal STA timing tools that perform incremental timing analysis as alternatives to ST. The UCLA timer [27] performs statistical as well as deterministic timing analysis and reports up to 31% reduction in runtime, as well as, high correlation, with respect to Monte Carlo simulations. Hu et al. [14] have developed a timer to aid in gate sizing for leakage optimization. They implement a fast "incremental" STA tool along with a full timing STA tool. Using the STA tool, they propose a gate sizing heuristic. When we implement their algorithm in C/C++, we observe that their incremental timing tool can have large errors with respect to a ST. Our work is relevant to works such as [14], in that we model wire delay and slew in an internal iSTA tool (using learning-based techniques) so as to achieve the slowest possible divergence from a ST.

Offset-based timing calibration has been introduced by Moon et al. [20]. The idea is to improve the accuracy of a given STA engine by periodically invoking a signoff timer and storing slack differences (offsets) at every timing endpoint. When the STA engine produces new timing estimates (e.g., during optimization), they are adjusted by slack offsets. Our analysis shows that if critical paths change due to sizing operations, applying offset-based endpoint slack correlation can lead to large errors. Our work is a significant improvement from [20] since we correlate other timing information such as slew, delay and AAT to the ST.

Machine learning approaches for delay estimation can be powerful since they capture complex interactions between design parameters (e.g., number of instances, number of primary inputs and outputs, hierarchy, etc.) and physical design contexts (e.g., parasitics, wirelength, buffer insertion, etc.). Gelosh and Setliff [10] propose a rule-based learning approach, similar to classification and regression trees, to capture parameter interactions to estimate area and delay. They report that their estimations can be optimistic with percentage errors up to 6.67%. Bao [5] presents a learning-based approach to estimate path slack by fitting the slack delta before and after Vt-swap for leakage power minimization. Worst-case error of 5.33ps is reported. From the description of [5], specific modeling parameters and the "amount of downstream logic" needed to represent the effect of slew propagation are unclear. In addition, missing testcase details (#cells, #nets, etc.) make it difficult to assess extensibility of the methodology and models to non-graphics contexts. These previous works report results on specific paths in a circuit and do not assess design-level endpoint slack metrics or how propagated errors are minimized along timing paths.

Machine learning has also been used in a variety of EDA applications. To model gate delay under process variation, Gao et al. [9] propose statistical gate delay modeling using artificial neural networks with dimension-order reduction of the parameter space. However, they report results only with random paths and not with real designs. Ganapathy et al. [8] capture effects of intra- and inter-die spatial variations and temporal fluctuations of temperature while fitting a regression model to estimate path delays. They validate their methodology on a 32KB cache, use SPICE simulations to train and test their models, and report a median error of less than 5%. Samanta et al. [22] perform wire and buffer sizing using Support Vector Machine regression to minimize variation on non-tree clock networks. They demonstrate up to 45% reduction in skew with their machine learning-based sizing algorithm.

III. Methodology

We now provide background on the analytical models that we study and a description of our offset-based modeling methodology.

A. Analytical Models for Wire Delay and Slew

We use well-known analytical models for wire delay and slew as a starting point to derive learning-based models.

Wire Delay. We use two models for wire delay – Elmore delay (EM) [6] and D2M [2]. Given a routing tree $T(N)$ rooted at source node n_0, Elmore delay is given by

$$EM = t(n_i) = r_d \cdot C_{n_0} + \sum_{e_v \in\ path(n_0,n_i)} r_{e_v} \cdot \left(\frac{c_{e_v}}{2} + C_v\right) \quad (1)$$

where $t(n_i)$ is the Elmore delay at node n_i, r_d is the output resistance of the driver at the net's source, C_{n_0} is the output pin capacitance of the driver at the net's source, e_v is the edge from node v to its parent in $T(N)$, r_{e_v} is the resistance of e_v, c_{e_v} is the capacitance of e_v, and C_v is the *tree capacitance* of the subtree rooted at node v.

Elmore delay is the first moment of impulse response and can be inaccurate when there is a high degree of resistive shielding [3]. Alpert et al. [2] propose the "delay with 2 moments" or D2M metric which is a simple function of the first two circuit moments, m_1 and m_2 respectively. The D2M delay metric is given by

$$D2M = \frac{m_1^2}{\sqrt{m_2}} \cdot \ln 2 \quad (2)$$

$$D2M_r = \alpha \cdot D2M + (1-\alpha)EM \quad (3)$$

$$\alpha = \left(\frac{2m_2 - m_1^2}{2m_2 - m_1^2 + T^2/12}\right)^{5/2} \quad (4)$$

where $D2M$ is the delay with two moments metric for step input, $D2M_r$ is the delay with two moments metric for ramp input, EM is the Elmore delay of the wire, α reflects the degree of significance of ramp input, and T is input slew of the wire.

Wire Slew. We consider three models for wire slew – *PERI* [17], scaled S2M [1], and Lognormal slew [3]. The *PERI* model for wire slew [4] is given by

$$PERI = \sqrt{T^2 + \ln 9 \cdot EM^2} \quad (5)$$

Agarwal et al. [1] propose scaled "slew with 2 moments" or S2M which is accurate for both near-end and far-end nodes. For a step input, $S2M_s$ is given by

$$S2M_s = \frac{\sqrt{-m_1}}{\sqrt[4]{m_2}} \cdot \ln 9 \cdot \left(\sqrt{2m_2 - m_1^2}\right) \quad (6)$$

The scaled S2M metric is extended to ramp inputs using the *PERI* technique [17] and is given by

$$S2M = \sqrt{{S2M_s}^2 + T^2} \quad (7)$$

Lognormal slew [3], derived from matching the first and second moments, is given for step input by

$$LNS_s = \frac{m_1^2}{\sqrt{2m_2}} \cdot \left(e^{k\cdot\sqrt{2\ln\left(\frac{2m_2}{m_1^2}\right)}} - e^{-k\cdot\sqrt{2\ln\left(\frac{2m_2}{m_1^2}\right)}}\right) \quad (8)$$

with $k \approx 0.9062$. It is extended to ramp inputs using the *PERI* technique as

$$LNS = \sqrt{{LNS_s}^2 + T^2} \quad (9)$$

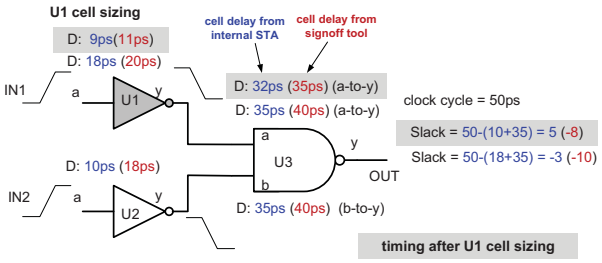

Fig. 2. Example of the offset-based timing correlation after cell sizing.

We study weighted combinations of the analytical models for wire delay and slew and determine the weights by using machine learning techniques. We also use values of α (from Equation (4)), T wire input slew, and $D2M_r$ as "classifiers", that is, their values determine the need for different models such that the maximum absolute error (MAE) as well as the average absolute error (AAE) are minimized. For wire delay, we choose $D2M$, $D2M_r$, EM, $\alpha \cdot D2M$ ($AD2M$), and $\alpha \cdot EM$ (AEM) as regressors and α, T and $D2M_r$ as classifiers. For wire slew, we choose $PERI$, $S2M_s$, $S2M$, LNS_s and LNS as regressors and α and T as classifiers. We exhaustively study combinations of {one, two, three} regressors $\times$ {zero, one, two} classifiers for wire delay and slew, and use this analysis to derive our best learning-based models (ML).

B. Correlating Timing Results with a Signoff STA Tool

To correlate internal timing with a ST, iSTA tools invoke the ST and obtain "golden" timing values from them. Moon et al. [20] propose an offset-based endpoint-slack correlation approach. They perform STA on both timers (signoff and internal), and compare the slack at each timing endpoint to compute the slack difference (*offset*). These offset values are used by the internal timer to compensate the timing discrepancy. However, after several gate sizing operations, the offset values are no longer accurate. The correlated slack can have large errors if the critical path changes after gate sizing.

Figure 2 shows an example of timing correlation after gate sizing (for simplicity, wire delay is ignored). In the example, red numbers and blue numbers are timing values respectively from a ST and an internal iSTA tool. Before gate sizing, the endpoint slack values respectively from the ST and internal tools are -10ps and -3ps. With slack correlation, the difference in slack is -7ps and is used as the slack offset. After sizing cell U1 (in the figure), the slack difference becomes -13ps, and the previous slack offset (-7ps) is not enough to compensate this new difference of -13ps. This discrepancy occurs since the critical path changes from $IN1-U1.y-OUT$ to $IN2-U1.y-OUT$. Due to scenarios such as this, the slack correlation method from [20] is neither sufficient nor robust.

To reduce such timing discrepancies after cell sizing, we implement different correlation approaches by correlating additional timing information such as transition time, cell and wire delay, and actual arrival time (AAT). STA (full or incremental) has four basic steps in its timing calculations: (1) slew, (2) delay, (3) actual arrival time (AAT) and required arrival time (RAT), and (4) slack calculation. We correlate the corresponding timing information with a ST at each step. In Figure 2, if we correlate cell delay values and save cell delay offsets, the slack discrepancy will disappear since the endpoint slack is recalculated based on the new critical path ($IN2-U1.y-OUT$).

Algorithm 1 presents pseudocode of our timing correlation procedure. The procedure saves offset values for each type of timing information from a ST. *Correlate_TR*(), *Correlate_DL*(), *Correlate_AAT*() and *Correlate_SLK*() are the correlation procedures respectively for transition time, delay, AAT/RAT and slack. *en_TR*, *en_DL*, *en_AAT* and *en_SLK* refer to enable/disable flags for each type of timing information. By using these flags, we control the timing information that is correlated in the internal iSTA tool.

The *Correlate_TR*() procedure obtains the transition time offsets (TR_OFS) for pins of each cell in a topological order. In the *Correlate_TR*() procedure, TR_ST_p and TR_IT_p respectively refer to transition times of pin p from ST and internal tool. The offset value, TR_OFS_p, is calculated for each pin as the difference between TR_ST_p and TR_IT_p (Lines 3 and 7 in *Correlate_TR*()). $TR\_IT_{p_j}$ of a cell input pin p_j is computed using a wire slew model (e.g., $PERI$, $S2M$, ML, etc.). $TR\_IT_{p_i}$ of a cell output pin p_i is computed from a gate slew model (e.g., LUT). After saving the offset value, we replace TR_IT_p with TR_ST_p for slew calculations in the next stages; to avoid error propagation in the offset, correlated input transition times are used for calculating slew. In the *Correlate_DL*() procedure, DL_ST_a and DL_IT_a respectively refer to delay values in arc a from a ST and internal tool. The offset value, DL_OFS_a for each arc a is obtained by calculating the difference between DL_ST_a and DL_IT_a (Line 4 in *Correlate_DL*()). DL_IT_a for cell delay is computed from the internal tool's gate delay model (e.g., LUT). DL_IT_a for wire delay is computed using a wire delay model (e.g., EM, $D2M$, ML, etc.). *Correlate_AAT*() and *Correlate_SLK*() procedures are implemented in a similar way as the *Correlate_TR*() procedure.

Algorithm 2 describes pseudocode of static timing analysis with the offset values. The *Calculate_TR*(), *Calculate_DL*(),

Algorithm 1 Offset-based timing correlation.

Procedure *Correlate*(N, *en_TR*, *en_DL*, *en_AAT*, *en_SLK*)
Input : netlist N, enable flags for each correlation
1: Initialize offset values as zero;
2: **if** *en_TR* **then**
3: *Correlate_TR*(N);
4: **end if**
5: **if** *en_DL* **then**
6: *Correlate_DL*(N);
7: **end if**
8: **if** *en_AAT* **then**
9: *Correlate_AAT*(N);
10: **end if**
11: **if** *en_SLK* **then**
12: *Correlate_SLK*(N);
13: **end if**

Procedure *Correlate_TR*(N)
Input : transition time, TR_ST_p of each pin p from ST
Output : transition time offset, TR_OFS_p of each pin p
1: **for all** cell instance c_i in the netlist N with a topological order **do**
2: **for all** input pin p_j in the cell instance c_i **do**
3: $TR\_OFS_{p_j} \leftarrow TR\_ST_{p_j} - TR\_IT_{p_j}$;
4: $TR\_IT_{p_j} \leftarrow TR\_ST_{p_j}$;
5: **end for**
6: $p_i \leftarrow$ output pin of c_i;
7: $TR\_OFS_{p_i} \leftarrow TR\_ST_{p_i} - TR\_IT_{p_i}$;
8: $TR\_IT_{p_i} \leftarrow TR\_ST_{p_i}$;
9: **end for**

Procedure *Correlate_DL*(N)
Input : delay, DL_ST of each (pin-to-pin) arc from ST
Output : delay offset, OFS of each arc a
1: **for all** primary port or pin, p_i in the netlist N **do**
2: **for all** sink pin p_j of the source pin p_i **do**
3: $a_{ij} \leftarrow$ arc from p_i to p_j;
4: $DL\_OFS_{a_{ij}} \leftarrow DL\_ST_{a_{ij}} - DL\_IT_{a_{ij}}$;
5: $DL\_IT_{a_{ij}} \leftarrow DL\_ST_{a_{ij}}$;
6: **end for**
7: **end for**

Calculate_AAT() and *Calculate_SLK*() procedures respectively calculate transition time, cell and wire delay, AAT/RAT, and slack using the saved offset values. In *Calculate_TR*(), TR_IT_p of a cell input pin and output pin is computed using a wire slew model and gate slew model, respectively. TR_p refers to compensated transition time of pin p with an offset TR_OFS_p (Lines 3 and 6 in *Calculate_TR*()). In *Calculate_DL*(), DL_IT_a for cell delay and wire delay are respectively computed from the internal tool's gate delay model and wire delay model. DL_a is the compensated delay of timing arc a. *Calculate_AAT*() and *Calculate_SLK*() are implemented in a similar way as the *Calculate_TR*() procedure.

Algorithm 2 Static timing analysis with offset values.

Procedure *iSTA*(N)
Input : netlist N, correlated offset values
1: *Calculate_TR*(N);
2: *Calculate_DL*(N);
3: *Calculate_AAT*(N);
4: *Calculate_SLK*(N);

Procedure *Calculate_TR*(N)
Input : transition time offset, TR_OFS_p of each pin p
Output : transition time TR_p of each pin p
1: **for all** cell instance c_i in the netlist N with a topological order **do**
2: **for all** input pin p_j in the cell instance c_i **do**
3: $TR_{p_j} \leftarrow TR\_IT_{p_j} + TR\_OFS_{p_j}$;
4: **end for**
5: $p_i \leftarrow$ output pin of c_i;
6: $TR_{p_i} \leftarrow TR\_IT_{p_i} + TR\_OFS_{p_i}$;
7: **end for**

Procedure *Calculate_DL*(N)
Input : delay offset, DL_OFS of each arc a
Output : delay DL_a of each arc a
1: **for all** primary input or cell input/output pin, p_i in the netlist N **do**
2: **for all** sink pin p_j of the source pin p_i **do**
3: $a_{ij} \leftarrow$ arc from p_i to p_j;
4: $DL_{a_{ij}} \leftarrow DL\_IT_{a_{ij}} + DL\_OFS_{a_{ij}}$;
5: **end for**
6: **end for**

We implement each timing correlation approach, and explore the impact of different correlation approaches when cells are changed through netlist perturbations such as would occur during gate sizing. Figure 3 illustrates a gate sizing flow with timing correlation. The timing correlation procedure invokes ST and saves offset values for each type of timing information. During gate sizing, the internal tool calculates timing information with the offset values. However, timing discrepancy with ST will increase as cells are changed due to gate sizing. To reduce (or control) the amount of timing discrepancy (or error), we correlate timing information when every N cells are changed.

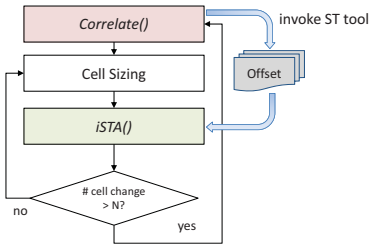

Fig. 3. Gate sizing flow incorporating offset-based timing correlation of an internal iSTA tool to the ST.

IV. Experimental Setup and Results

We conduct three experiments that assess our modeling and offset-based correlation methods (Experiments 1, 2) and evaluate them in our motivating use context of an internal iSTA tool (Experiment 3).

- Experiment 1: accuracy of learning-based interconnect models.
- Experiment 2: impact of correlation methodology on timing discrepancy.
- Experiment 3: impact of update accuracy.

We use four benchmarks, namely, *pci_bridge32, fft, matrix_mult* and *edit_dist* from the ISPD 2013 gate sizing contest [26] as our test circuits. Table I shows statistics of the benchmarks. We use the ISPD 2013 contest timing library, and our ST is a commercial signoff STA tool used in the ISPD 2013 contest. For gate delay and slew, we apply interpolation and extrapolation of values in the Liberty LUTs in the timing library.

TABLE I
Benchmark statistics (PI = Primary Input; PO = Primary Output).

Benchmark	#cells	#nets	#PI	#FFs	#pins	#PO
pci_bridge32	30603	30763	160	3359	87813	201
fft	32766	33792	1026	1984	105355	1984
matrix_mult	156440	159642	3202	2898	459946	1600
edit_dist	126665	129227	2562	5661	374606	12

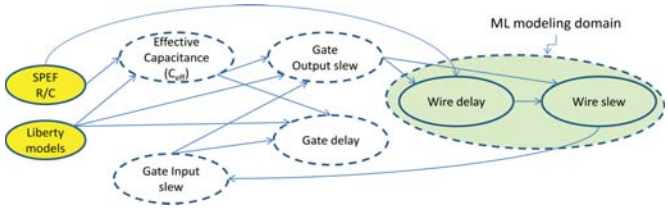

Fig. 4. Flow graph of modeling parameters to estimate circuit delay.

A. Experiment 1: Accuracy of Learning-Based Interconnect Models

We derive wire delay and slew models by using least-squares regression (LSQR) [18] to fit values from the ST. We generate training samples from benchmark netlists that have no slack violations and contain heterogeneous mixes of cell sizes and types. From the ST, we obtain delay and slew at every pin in the netlist, and fit our models to the data from ST. We use 50% of these data points for training, derive models using LSQR, test the models on all data points, and compute the estimation errors. Figure 4 shows a flow graph of parameters required to estimate circuit delay.

Exhaustive search for the best regressor(s) and classifier(s)

We follow the methodology described in Section III to determine the best choice of regressors and classifiers in learning-based models for wire delay and slew. We sweep values of α from zero to one, $D2M_r$ from 10ps to 200ps, and T from 20ps to 250ps. Figure 5 shows results for modeling of wire delay using exhaustive search with up to three regressors and two classifiers. We observe the following.

1 With one regressor and no classifier (**1R0C**), MAE of the best model is 20ps. The best model uses $D2M_r$ as the regressor.
2 With two regressors and one classifier (**2R1C**), MAE of the best model reduces to 16ps. The best model uses $D2M$ and $D2M_r$ as regressors and $\alpha = 0.94$ as the classifier.
3 With three regressors and one classifier (**3R1C**), MAE of the best model is 14ps. The best model uses EM, $AD2M$ and AEM as regressors and $\alpha = 0.94$ as the classifier.

4 With three regressors and two classifiers (**3R2C**), MAE and AAE do not improve significantly. The best model uses *EM*, *AD2M* and *AEM* as regressors and $\alpha = 0.1$ and $\alpha = 0.94$ as classifiers. We stop our search for wire delay models at this stage.

Figure 6 shows results for modeling of wire slew using exhaustive search with up to three regressors and two classifiers. We observe the following.

1 With one regressor and no classifier (**1R0C**), MAE is 73ps. The best model uses PERI as the regressor.
2 With two regressors and two classifiers (**2R2C**), MAE reduces to 32ps. The best model uses EM^2 and LNS_s^2 as regressors and $\alpha = 0.96$ and $\alpha = 0.99$ as classifiers.
3 With three regressors and one classifier (**3R1C**), MAE is 33ps. The best model uses EM^2, LNS_s^2 and T^2 as regressors and $\alpha = 0.96$ as the classifier.
4 With three regressors and two classifiers (**3R2C**), MAE reduces to 31.5ps. The best model uses EM^2, LNS_s^2 and T^2 as regressors and $\alpha = 0.7$ and $\alpha = 0.96$ as classifiers. We stop our search at this stage since the complexity of models becomes high.

We use these models in our internal iSTA tool, and Table II shows endpoint slack AAE compared to ST without timing correlation. We observe that three regressors and two classifiers (**3R2C**) for wire delay, and two regressors and two classifiers (**2R2C**) for wire slew, together give the minimum endpoint slack AAE.

TABLE II
ENDPOINT SLACK AAE FOR EACH COMBINATION OF DELAY, SLEW MODELS.

Model		AAE in slack (ps)			
Delay	Slew	pci_bridge32	fft	edit_dist	matrix_mult
1R0C	1R0C	15.25	41.87	144.40	39.70
1R0C	2R2C	15.80	47.43	158.97	43.74
1R0C	3R1C	16.36	47.87	160.88	44.85
1R0C	3R2C	16.12	47.81	161.36	44.71
2R1C	1R0C	16.60	44.61	158.33	46.79
2R1C	2R2C	17.16	50.01	172.86	50.69
2R1C	3R1C	17.73	50.55	174.78	51.80
2R1C	3R2C	17.48	50.40	175.23	51.65
3R1C	1R0C	17.10	49.41	174.92	50.44
3R1C	2R2C	17.83	55.83	191.57	54.91
3R1C	3R1C	18.41	56.41	193.59	56.05
3R1C	3R2C	18.18	56.28	194.14	55.95
3R2C	1R0C	7.24	17.23	47.55	11.02
3R2C	2R2C	7.44	16.08	44.12	10.90
3R2C	3R1C	7.45	16.13	44.06	11.09
3R2C	3R2C	7.53	16.16	44.12	11.02

Best learning-based models for wire delay and slew.

Our best learning-based model for wire delay is a weighted combination of *AEM*, *AD2M* and *EM* and is given by

$$WD_{ML} = \begin{cases} a_1 \cdot AEM + a_2 \cdot AD2M + a_3 \cdot EM, & \text{if } \alpha < 0.3 \\ b_1 \cdot AEM + b_2 \cdot AD2M + b_3 \cdot EM, & \text{if } 0.3 \le \alpha < 0.94 \\ c_1 \cdot AEM + c_2 \cdot AD2M + c_3 \cdot EM, & \text{if } \alpha \ge 0.94 \end{cases} \quad (10)$$

where *EM* is computed from Equation (1), *D2M* is computed from Equation (2), α is computed from Equation (4), *AEM* is $\alpha \cdot EM$ and *AD2M* is $\alpha \cdot D2M$. $a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3$ are the regression coefficients, their values are shown in Table III. The AAE is 0.23ps and MAE is 14.25ps. This model applies to both rising and falling transitions of signals.

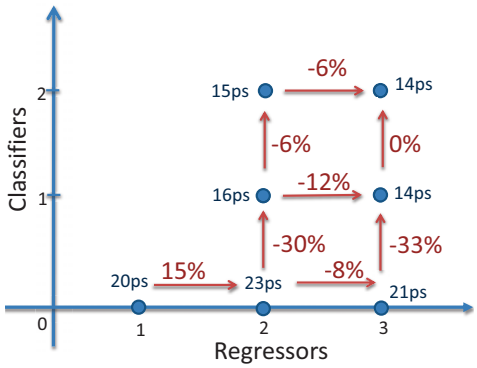

Fig. 5. Wire delay model MAE with different numbers of regressors and classifiers.

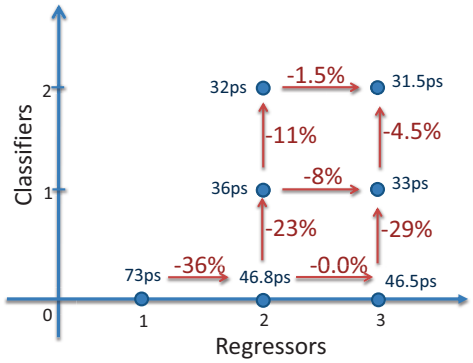

Fig. 6. Wire slew model MAE with different numbers of regressors and classifiers.

TABLE III
LEARNING-BASED WIRE DELAY MODEL COEFFICIENTS.

a_1	a_2	a_3	b_1	b_2	b_3	c_1	c_2	c_3
-1.72	2.35	0.96	-1.05	1.27	1.02	-2.72	1.41	2.50

Our best learning-based model for wire slew is a weighted combination of EM^2 and LNS_s^2 and is given by

$$WS_{ML} = \begin{cases} \sqrt{a_1 \cdot EM^2 + a_2 \cdot LNS_s^2 + T^2}, & \text{if } \alpha < 0.96 \\ \sqrt{b_1 \cdot EM^2 + b_2 \cdot LNS_s^2 + T^2}, & \text{if } 0.96 \le \alpha < 0.99 \\ \sqrt{c_1 \cdot EM^2 + c_2 \cdot LNS_s^2 + T^2}, & \text{if } \alpha \ge 0.99 \end{cases} \quad (11)$$

where LNS_s is computed from Equation (8), T is input slew of the wire, and $a_1, a_2, b_1, b_2, c_1, c_2$ are the regression coefficients. Values of these regression coefficients are shown in Table IV. The AAE is 0.23ps and MAE is 32.77ps. This model applies to both rising and falling transitions of signals.

TABLE IV
LEARNING-BASED WIRE SLEW MODEL COEFFICIENTS.

a_1	a_2	b_1	b_2	c_1	c_2
-3.44	2.07	-2.39	1.59	-1.88	1.30

(i) Accuracy of estimation in individual timing arcs. Our learning-based models are more accurate than previous analytical models in part because they can capture some of the "black-box" modeling of the ST in timing paths. Our studies indicate that gate delay and slew estimations based on the Liberty LUT are fairly accurate with respect to the ST. We also estimate effective load capacitance using the model from [19] and observe that the error with respect to ST

is very small. Therefore, we do not derive learning-based models for gate delay, slew and effective load capacitance. Figures 7 and 8 respectively show probability density functions (PDFs) of error (i.e., the difference between model estimates and ST estimates) for wire slew and delay across the four benchmarks listed in Table I. Our learning-based models (ML) minimize the divergence of wire delay and slew at each pin from the ST values. The models also bring the mean of the error distribution close to zero. ML estimates are quite accurate as they have a narrow distribution with small σ for both wire delay and slew. Elmore delay has the largest spread in error for wire delay, and S2M has the largest spread in error for wire slew.

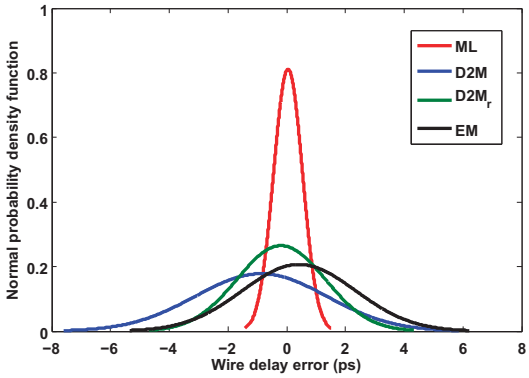

Fig. 7. PDF of wire delay estimation error for WD_{ML} (*ML* in figure), *D2M*, $D2M_r$ and *EM*.

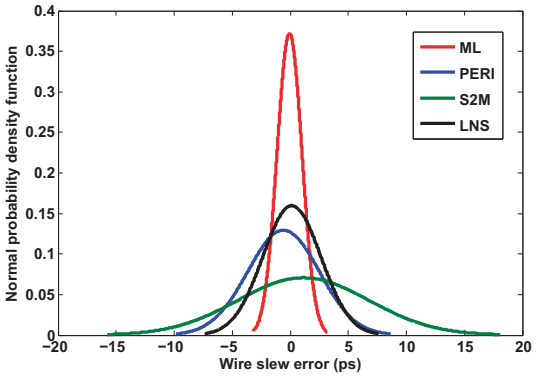

Fig. 8. PDF of wire slew estimation error for WS_{ML} (*ML* in figure), *PERI*, *S2M*, and *LNS*.

(ii) Accuracy of estimation in timing paths. We change 2.5% – 30% of the cells in each benchmark and study the accuracy impact of ML models in estimating endpoint slack. Figure 9 shows endpoint slack MAE and AAE for the *fft* and *matrix_mult* benchmarks when AAT and RAT on timing arcs are correlated with ST. Figure 10 shows endpoint slack MAE and AAE for the *fft* and *matrix_mult* benchmarks when no offset-based correlation is applied. In the figures, combinations of delay and slew models are listed as (*delay, slew*) tuples. Among the ML models, (D2M, ML) has the smallest errors. If, e.g., endpoint slack MAE of $\sim$10ps is tolerated, (D2M, ML) allows correlations with ST to compute offsets to be delayed until $\sim$10% of cells are changed.

Table V shows the maximum and average absolute errors in slack estimation of various combinations of analytical and ML models across all benchmarks. From the results, (D2M, ML) shows accurate estimates of the endpoint slack in *pci_bridge32* and *fft*.

TABLE V
ENDPOINT SLACK ERRORS WITH DIFFERENT (DELAY, SLEW) MODELS AND WITH TIMING CORRELATION.

Model	MAE			AAE		
	% cells changed			% cells changed		
(Delay, Slew)	5	15	30	5	15	30
pci_bridge32						
(EM, PERI)	3.529	5.482	8.981	0.291	0.601	1.114
(D2M, PERI)	3.680	5.173	9.205	0.230	0.513	1.024
(D2M, ML)	3.127	4.404	8.968	0.234	0.522	0.979
(ML, S2M)	9.515	6.820	9.327	0.259	0.605	1.095
(ML, PERI)	4.072	5.867	10.580	0.246	0.568	1.175
(ML, ML)	3.485	5.459	9.497	0.246	0.565	1.113
fft						
(EM, PERI)	10.589	13.192	14.313	0.770	1.451	1.387
(D2M, PERI)	10.454	10.704	11.203	0.472	0.891	0.881
(D2M, ML)	8.348	9.518	10.245	0.514	0.946	0.670
(ML, S2M)	14.208	18.856	24.117	0.829	1.351	1.242
(ML, PERI)	10.803	14.393	13.765	0.689	1.202	1.194
(ML, ML)	8.956	11.718	13.190	0.745	1.262	1.119
edit_dist						
(EM, PERI)	14.685	21.625	68.538	2.697	3.816	7.408
(D2M, PERI)	12.452	18.997	50.984	0.735	2.519	3.788
(D2M, ML)	14.811	17.842	51.325	1.071	2.740	4.058
(ML, S2M)	27.161	34.429	54.182	2.456	3.599	6.234
(ML, PERI)	27.067	34.588	53.378	2.383	3.623	6.065
(ML, ML)	28.186	35.250	53.736	2.466	3.650	6.030
matrix_mult						
(EM, PERI)	7.515	16.769	19.277	0.684	1.822	2.359
(D2M, PERI)	5.518	10.038	12.602	0.318	0.885	1.231
(D2M, ML)	6.226	11.428	14.018	0.401	1.083	1.402
(ML, S2M)	19.171	15.683	16.918	0.674	1.239	1.695
(ML, PERI)	17.317	13.783	21.638	0.650	1.368	1.730
(ML, ML)	18.282	15.053	18.893	0.658	1.269	1.692

B. Experiment 2: Correlation Methodology Impact on Timing Discrepancy.

In this experiment, we study the impact of different correlation approaches on the divergence in endpoint slack values from the ST. As described in Section III-B, we may apply offset-based correlation for different types of timing information. Specifically, we independently correlate (a) delay, (b) actual arrival time (AAT) and (c) slack values. We also correlate transition time because doing so can reduce the magnitude of offset values for delay and AAT.

Figure 11 compares endpoint slack MAE with different correlation approaches. We observe that only slack correlation (SLK in Figure 11) is not sufficient and can lead to large MAE in endpoint slack estimation. However, correlating actual arrival times (AAT in Figure 11) reduces the timing discrepancy since the AAT offset has more information than the slack offset, which considers only the timing of a critical path. Delay correlation (DLY in Figure 11) can improve the results further, since it is based on delay information for each arc; AAT correlation is based only on delay of the worst arc. Transition time correlation (TRAN in Figure 11) has smaller MAE when it is applied with AAT correlation. Delay and AAT correlation provide a significant improvement of endpoint slack error over the slack correlation approach from [20], e.g., the error after 5% cell changes is reduced from 105ps to 7ps for *fft*.

In addition, frequency of correlation affects the endpoint slack error. There is a tradeoff between runtime and accuracy depending on the frequency of correlation. Table VI shows the tradeoff between runtime and accuracy in *edit_dist* and *matrix_mult* for different correlation frequencies that correspond to the percentages of cells changed. The runtime is normalized to the case of correlation after every 5% changed cells.

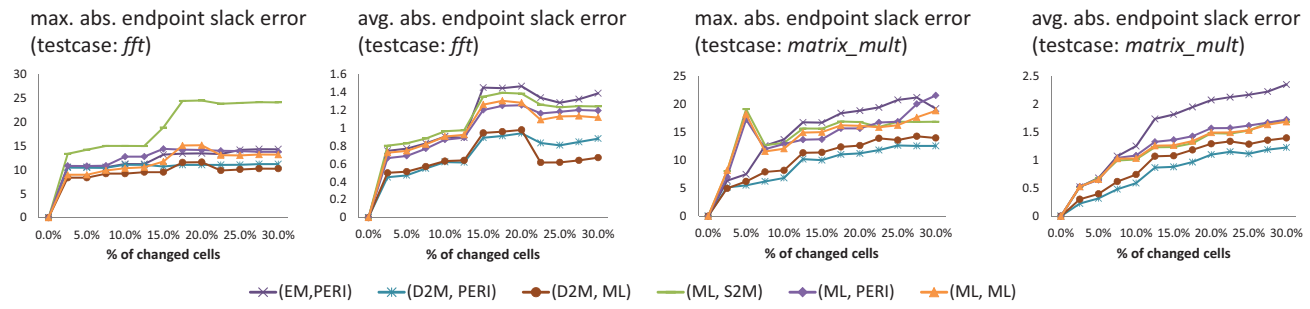

Fig. 9. Endpoint slack error after correlating with a ST (MAE and AAE).

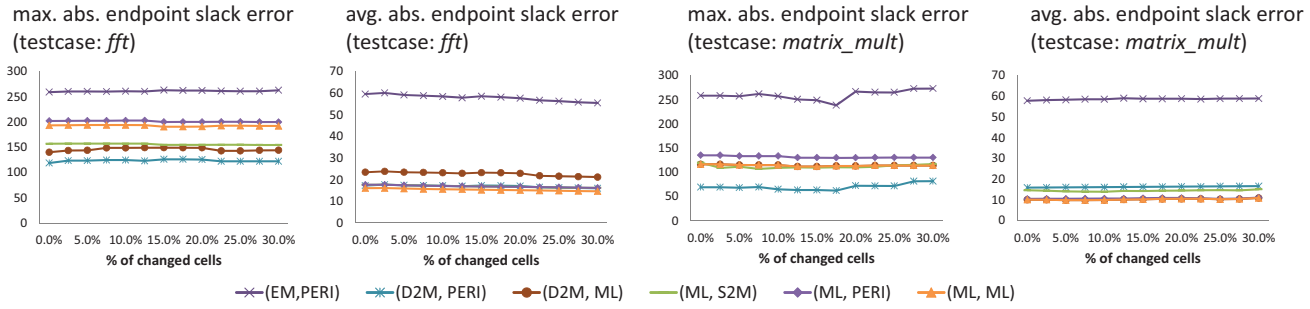

Fig. 10. Endpoint slack error without correlating with a ST (MAE and AAE).

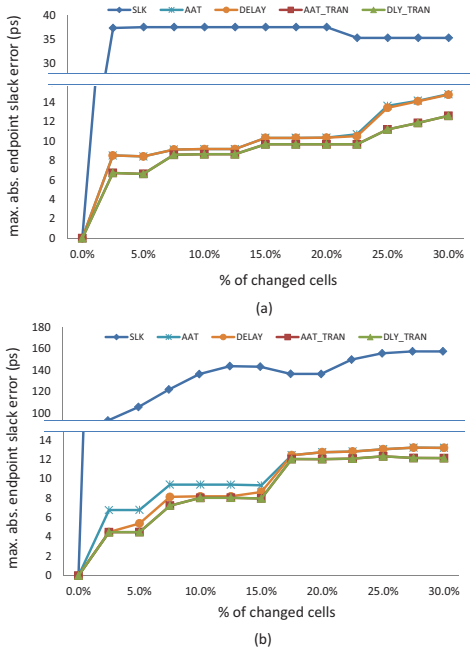

Fig. 11. Results of correlation vs. quality study on (a) *pci_bridge32* and (b) *fft*.

TABLE VI
TRADEOFFS BETWEEN RUNTIME AND ACCURACY AT DIFFERENT CORRELATION FREQUENCIES.

Correlation frequency	*edit_dist*		*matrix_mult*	
	AAE (ps)	normalized runtime	AAE (ps)	normalized runtime
5%	2.37	1.00	0.63	1.00
10%	2.54	0.85	1.13	0.84
15%	4.43	0.78	1.26	0.79
30%	6.25	0.69	1.64	0.73

C. Experiment 3: Impact of Update Accuracy.

We assess accuracy of our internal iSTA tool by studying impact of update accuracy on endpoint slack. We change sizes and/or Vt of 10% of cells and then revert them to their original states. We repeat this experiment for different values of iSTA margin.[1] After reverting the changed cells to their original states, we recalculate timing values on each pin. The error is the difference between the recalculated timing values and original timing values. Figure 12 shows the maximum endpoint slack error when 10% of the cells are changed for all benchmarks. From the results, when iSTA margin is small (less than 2ps), endpoint slack MAE is negligible. However, when the iSTA margin is large, endpoint slack errors increase because we do not propagate small timing changes into next stages.

This experiment, whose structure recalls the study of incremental optimization in [15], raises an open issue of how tight the update accuracy must be to prevent divergence even in cases where we know (by our construction of the experiment) that the delta should be zero.

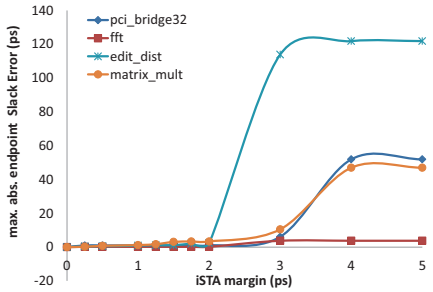

Fig. 12. Endpoint slack MAE with 10% cells changed in iSTA.

[1]iSTA margin is a threshold that determines whether delay and slew calculations should be propagated to the next (or previous) stages in the timing path. Timing values are propagated only if the change is greater than or equal to the value of the iSTA margin.

V. Conclusions and Future Work

Incremental STA tools are important for iterative sizing and Vt-swapping heuristics. In this work, to model wire delay and slew in an internal iSTA tool, we explore several analytical and machine learning-based models in conjunction with offset-based timing correlation. The ML-based models can slow the deviation in endpoint slack from a ST, and thus reduce the number of calls to the ST tool to perform the timing correlation. We show that our ML models are accurate in estimating wire delay and slew for individual timing arcs. When used with a combination of analytical models, the ML models (especially (D2M, ML)) can be accurate in estimating endpoint slack and path-based delay. Experiments on offset-based timing correlation further show that our methodology has up to 10x less endpoint slack estimation error than offset-based slack correlation [20].

Our studies with iSTA margin for propagation of delay/slew calculation in the timing graph raises an open issue of how tight update accuracy must be to prevent divergence even in cases where we know that the error should be zero. Our ongoing work studies learning-based models to minimize maximum error for path-based (as opposed to stage-based) delay modeling, as well as alternative methods for offset correlation in the timing graph.

Acknowledgments

We gratefully acknowledge research support from NSF, MARCO/DARPA, IMPACT, Qualcomm and the Semiconductor Research Corporation.

References

[1] K. Agarwal, D. Sylvester and D. Blaauw, "A Simple Metric for Slew Rate of RC Circuits Based on Two Circuit Moments", *IEEE Trans. CAD* 23(9) (2004), pp. 1346-1354.

[2] C. J. Alpert, A. Devgan and C. Kashyap, "A Two Moment RC Delay Metric for Performance Optimization", *Proc. ISPD*, 2000, pp. 73-78.

[3] C. J. Alpert, F. Liu, C. Kashyap and A. Devgan, "Delay and Slew Metrics Using the Lognormal Distribution", *Proc. DAC*, 2003, pp. 382-385.

[4] H. B. Bakoglu, *Circuits, Interconnects, and Packaging for VLSI.* Reading, MA: Addison-Wesley, 1990.

[5] S. Bao, "Optimizing Leakage Power using Machine Learning", *CS229 Final Project*, Stanford University, 2010.

[6] W. C. Elmore, "The Transient Response of Damped Linear Network with Particular Regard To Wideband Amplifiers", *Journal of Applied Physics* 19 (1948), pp. 55-63.

[7] J. P. Fishburn and A. E. Dunlop, "Tilos: A Posynomial Programming Approach to Transistor Sizing", *Proc. ICCAD*, 1985, pp. 326-328.

[8] S. Ganapathy, R. Canal, A. Gonzalez and A. Rubio, "Circuit Propagation Delay Estimation Through Multivariate Regression-Based Modeling Under Spatio-Temporal Variability", *Proc. DATE*, 2010, pp. 417-422.

[9] M. Gao, Z. Ye, Y. Wang and Z. Yu, "On Modeling the Digital Gate Delay Under Process Variation", *Journal of Semiconductors* 32(7) (2011), pp. 1-9.

[10] D. S. Gelosh and D. E. Setliff "Deriving Efficient Area and Delay Estimates by Modeling Layout Tools", *Proc. DAC*, 1995, pp. 402-407.

[11] P. Gupta, A. B. Kahng and P. Sharma, "A Practical Transistor-Level Dual Threshold Voltage Assignment Methodology", *Proc. ISQED*, 2005, pp. 421-426.

[12] P. Gupta, A. B. Kahng, P. Sharma and D. Sylvester, "Gate-Length Biasing for Runtime-Leakage Control", *IEEE Trans. CAD* 25(8) (2006), pp. 1475-1485.

[13] S. Hu, C. J. Alpert, J. Hu, S. K. Karandikar, Z. Li, W. Shi and C. N. Sze, "Fast Algorithms for Slew-Constrained Mininum-Cost Buffering", *IEEE Trans. CAD* 26(11) (2007), pp. 2009-2022.

[14] J. Hu, A. B. Kahng, S. Kang, M. Kim and I. Markov, "Sensitivity-Guided Metaheuristics for Accurate Discrete Gate Sizing", *Proc. ICCAD*, 2012, pp. 233-239.

[15] A. B. Kahng and S. Mantik, "On Mismatches Between Incremental Optimizers and Instance Perturbations in Physical Design Tools", *Proc. ICCAD*, 2000, pp. 17-21.

[16] A. B. Kahng, K. Masuko and S. Muddu, "Analytical Delay Models for VLSI Interconnects Under Ramp Input", *Proc. ICCAD*, 1996, pp. 30-36.

[17] C. V. Kashyap, C. J. Alpert, F. Liu and A. Devgan, "PERI: A Technique for Extending Delay and Slew Metrics to Ramp Inputs", *Proc. TAU*, 2002, pp. 57-62.

[18] S. S. Kozat and A. C. Singer, "Universal Switching Linear Least Squares Prediction" *IEEE Trans. Signal Processing* 56(1) (2008), pp. 189-204.

[19] S. P. McCormick, "Modeling and Simulation of VLSI Interconnects with Moments", *PhD Thesis*, MIT, June 1989.

[20] C. Moon, P. Gupta, P. J. Donehue and A. B. Kahng, "Designing a Digital Circuit by Correlating Different Static Timing Analyzers", *U.S. Patent* No. 7,823,098, 2010.

[21] M. M. Ozdal, S. Burns and J. Hu, "Gate Sizing and Device Technology Selection Algorithms for High-Performance Industrial Designs", *Proc. ICCAD*, 2011, pp. 724-731.

[22] R. Samanta, J. Hu and P. Li, "Discrete Buffer and Wire Sizing for Link-Based Non-Tree Clock Networks", *IEEE Trans. VLSI* 18(7) (2010), pp. 1025-1035.

[23] K. Shinkai, M. Hashimoto and T. Onoye, "A Gate-Delay Model Focusing on Current Fluctuation Over Wide Range of Process-Voltage-Temperature Variations", *Integration, the VLSI Journal* (2013), pp. 1-14.

[24] D. Sylvester and C. Hu, "Analytical Modeling and Characterization of Deep-Submicrometer Interconnect", *Proc. IEEE* 89(5) (2001), pp. 634-664.

[25] A. Chirayu, Intel Corp., *Personal communication*, January 2013.

[26] ISPD 2013 Discrete Gate Sizing Contest and Benchmark Suite. *http://ispd.cc/contests/13/ispd2013_contest.html*

[27] UCLA Timer. *http://nanocad.ee.ucla.edu/Main/Sizing*