Accounting for Floorplan Irregularity and Configuration Dependence in FPGA Routing Delay Models

Gabriel Barajas
Microchip
San Jose, California USA
gabriel.barajas@microchip.com

Jonathan W. Greene
Cambios Computing LLC
Palo Alto, California USA
jgreene@cambioscomputing.com

Fei Li
Microchip
San Jose, California USA
fei.li@microchip.com

James Tandon
California State University East Bay
Hayward, California USA
james.tandon@csueastbay.edu

***Abstract*—Accurate delay estimates for a user application implemented in a Field-Programmable Gate Array (FPGA) are essential for a quality FPGA timing flow and to avoid leaving performance on the table. FPGA inter-cluster routing consists of wire segments of a limited number of types which repeat in a somewhat regular pattern, interconnected by configurable muxes. The delay at any fanout of a segment can be significantly impacted by configuration-dependent capacitive loading related to other fanouts. Also, the insertion of RAM and math blocks into the FPGA floorplan introduces irregular stretching of the wire segments, altering their delays. We explain why and how commercial FPGA software typically employs a parameterized model for the delay at each fanout of a segment, based on the configuration and the irregularities present, with the parameters determined by fitting SPICE simulation data for a representative sample of cases. We propose incorporating readily-computed common path resistance values into the model. This enables high accuracy with fewer parameters and without the large amounts of SPICE data that would otherwise be required to explore interactions between floorplan irregularities and the set of active fanouts. In combination with other features of our models, errors in segment delay are reduced by almost half.**

Keywords— Field-programmable gate arrays, FPGAs, delay estimation, Elmore delay, common path resistance

I. Introduction

Field-Programmable Gate Arrays (FPGAs) are a widely-used form of programmable logic. A user application is implemented in an FPGA by configuring programmable logic blocks and routing switches or muxes. The user relies on FPGA timing software to guarantee the implementation can be safely clocked at the desired frequency, without risk of timing errors. This requires getting conservative estimates of the propagation delays through the FPGA circuitry as a function of the routing configuration. But if the estimates are unduly conservative, performance suffers. Thus, high-quality delay estimates are key.

In older FPGAs, the switches are implemented by NMOS pass transistors or anti-fuses, and can be adequately modeled as linear resistors. An RC tree may be constructed for each delay stage based on the relevant active portions of the parasitic netlist of the FPGA layout. (By active, we mean the portions reachable through switches that the configuration turns on.) The estimated delay to each active leaf (fanout) of the tree is then computed using a first-order model such as Elmore delay [1]. VPR [2] provides a widely-used example of this approach. Other, more accurate higher-order alternatives such as AWE [3] can be used instead of Elmore delay.

Since the advent of 0.13um technology brought a higher ratio of threshold to operating voltage (VT/VDD), "direct-drive" FPGA routing [4] is now preferred. In this style of routing there are at most two or three levels of switches, and switches are present only downstream of most of the wire capacitance, near the leaves of the delay tree. As a result, the delays to the various leaves of the tree become more similar to each other and less dependent on which other leaves are active. (In the extreme, one might be tempted to approximate the delays at all leaves with the same single value, independent of the configuration.)

As process technology continues to advance, transistors can no longer be approximated as linear resistors. The use of an accurate but time-consuming circuit simulator such as SPICE becomes necessary [5]. Nevertheless, we can still rapidly obtain accurate delay estimates for each routed FPGA application as follows:

1. Run SPICE simulation on a representative sample of routing trees. The sample includes various types of trees with various subsets of their leaves active. The simpler nature of direct-drive architectures helps to limit the number of samples required.
2. Fit some kind of regression model to that data.
3. Given a particular configuration, use the model to estimate the delay to each active leaf of each tree.

The first two steps, especially SPICE simulation, can be very time-consuming. But these only need to be done once for each new FPGA architecture, not repeated during the analysis of each application.

The regression model may be a table lookup, a concise linear or non-linear equation, or some combination. Machine learning has also been introduced for this purpose, either to get rough delay estimates based on information available prior to routing [6] or more accurate estimates after routing [7].

The regression models used to support commercial FPGAs are typically proprietary. However, they are known to face two challenges.

The first is fanout dependence. Although direct-drive architectures reduce the fanout dependence of delays, they do not eliminate it. In fact, fanout dependence becomes more

To appear, 2023 24th Int'l Symp. on Quality Electronic Design (ISQED) https://doi.org/10.1109/ISQED57927.2023.10129290

significant in FinFET technologies due to the increase in gate capacitance relative to wire and junction capacitance [8]. For best accuracy (to avoid unnecessary delay margin) the regression model must depend on the quantity of active fanouts and perhaps even the particular subset of fanouts that are active (which we call *configuration dependence*).

The second challenge is irregularities in the FPGA's floorplan, which cause "discontinuities" [9] in delay as a function of the length of the routing tree. If the floorplan of the FPGA is a regular grid of identical logic tiles (also known as *clusters*, *LABs* or *CLBs*), delays may increase smoothly with the number of tiles spanned by a tree. At worst, the number of types of trees which must be simulated is small due to various symmetries. However modern large FPGAs are heterogeneous and their grid is interrupted by the insertion or substitution of other tiles such as math blocks, memory blocks, clock distribution, etc. (See Fig. 1.) To avoid a combinatorial explosion in the complexity of the model (and hence the number of trees that must be simulated to determine its parameters), the model may treat the irregularities as adjustments to the delay value of an uninterrupted tree. (See, for example, [9].)

To make matters worse, these two challenges interact. That is, the incremental delay at one fanout caused when activation of another fanout adds capacitive loading will depend on what irregularities are present and where.

Note that these difficulties are particular to the inter-tile delays, making them especially problematic to model. Unfortunately, inter-tile delays also comprise a significant fraction of total critical path delay, about 45 per cent and often more. So it is essential to model them accurately.

The goal of this paper is to find a way that regression models can accurately estimate inter-tile routing delays in the presence of both configuration dependence and floorplan irregularities while remaining terse enough to be trained with limited SPICE data and avoid over-fitting. To this end, we strive (for instance) to avoid having any model parameter that is dependent on both the configuration and the irregularities that are present. We show that this goal can be accomplished using a novel semi-empirical approach that combines parameter fitting with the Elmore delay formula.

II. Problem Formulation

We assume an FPGA is composed of instances of a limited number of *tiles*, and that all connections between tile instances are made by abutment.

A *delay tree* (or *tree* for short) is shown in Fig. 2. At the root of the tree is a *driver* (buffer or logic cell), in this case shown in the leftmost tile instance. Its output signal may propagate through various muxes M_i to various *leaves* (fanouts) L_i in the same or other tile instances. We assume any leaf downstream of a mux must be in the same tile as the mux (in other words no unbuffered signal can pass between tiles). The tree is topological in the sense that the exact relative physical positions of the tile instances are unspecified. The tree includes only those tile instances that contain the driver, muxes or leaves, not other tile instances through which the signal may need to pass on its way.

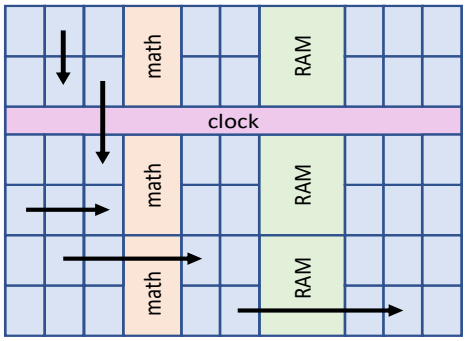

Fig. 1. Examples of discontinuities in a heterogeneous FPGA array. Each unlabeled blue tile contains logic and routing. A length one vertical connection may get stretched across a clock stripe. A length two horizontal connection may get stretched across math or RAM blocks. The stretch can occur at either the first or second half of the connection and cross either the top or bottom of the block.

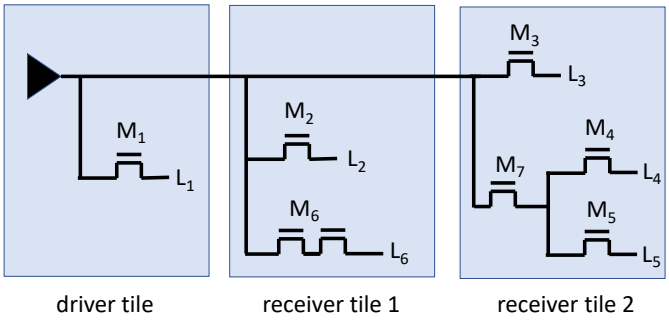

Fig. 2. Example of a delay tree containing three tile instances. L_1 – L_6 are the leaves that may be driven by the tree. M_1 – M_5 are the portions of single-stage muxes driving L_1 – L_5, respectively. M_4 and M_5 are in turn driven by a single-stage unbuffered mux M_7. M_6 is the portion of a two-stage mux driving L_6. Here we show switches as NMOS pass gates, but they can also be implemented by CMOS transmission gates [10], flash memory switches [11], antifuses [12], or anything similar that can be simulated by SPICE.

A *configuration* of a tree is specified by the subset of the muxes in the tree whose relevant input is selected.

A *polyomino* is a contiguous set of tile instances of specified types and at fixed relative positions. One of the tile instances is distinguished as the driving tile instance. Fig. 3 shows some example polyominos. Polyominos are the means by which we capture the impact of floorplan irregularities on a tree. The polyomino in which a tree is implemented determines which tile implements each portion of the tree, and each tile has a defined layout. Thus, the polyomino determines the complete, detailed parasitic netlist of the tree, including any stretching of wires over inserted tiles.

The independent variables available to our regression models are as follows:

- T is a tree.
- P is a polyomino which contains an implementation of the tree.
- $M_1, M_2, \ldots$ are the set of muxes in the tree.
- L is a leaf (fanout) of the tree whose delay is to be estimated.
- F is the signal transition: 1 if falling, 0 if rising.

- $R(P, M, L)$ is the resistance of the common portion of the paths to mux M and leaf L through tree T implemented in polyomino P.
- $X(M)$ is 1 if mux M is active, or 0 otherwise.
- $C = \{M_i : X(M_i) = 1\}$ is the set of active muxes in the tree (i.e., its configuration).

The dependent variable to be estimated is $D(P, L, F, C)$, the delay through tree T implemented in polyomino P from the input of the driver to leaf L with signal transition F when the tree's muxes are in configuration C.

The concept of common path resistance $R(P, M, L)$ may require some further explanation. Fig. 4 shows an example. The R values can be determined in a straightforward way by tracing paths through the netlists (including layout parasitics) of the tiles in the relevant polyomino P. (There is no need to run SPICE for this purpose.) The resistance values are independent of the particular user application, and thus may be tabulated once offline and used repeatedly for any application.

III. Methods

Recall that our goal is to find a good model for inter-tile delays in FPGAs, which (as explained in Section 1) are particularly difficult to estimate, yet account for almost half of total critical path delay. We propose a variety of such models and evaluate their relative merits in a simple and direct manner by determining how closely they can reproduce the corresponding SPICE delays.

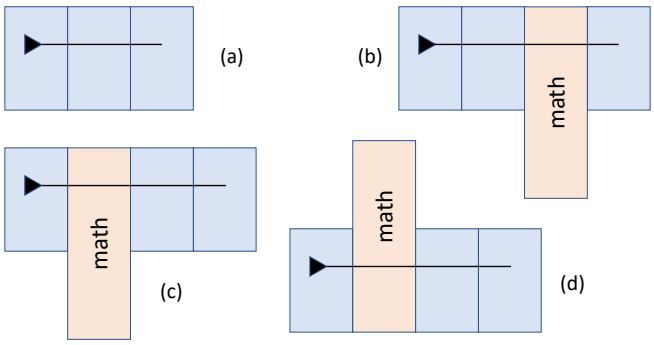

Fig. 3. Four examples of polyominos which may implement the tree of Figure 2. Each polyomino consists of a set of contiguous tile instances of specified types at specified relative positions. One logic/routing tile in each polyomino is distinguished as the driving tile (indicated here by the buffer symbol). Polyominos (a) and (b) differ because in (b) the wiring of the tree is stretched across the math block. Polyominos (c) and (d) differ because the wiring across the top of the math block may have different parasitics than wiring across the bottom.

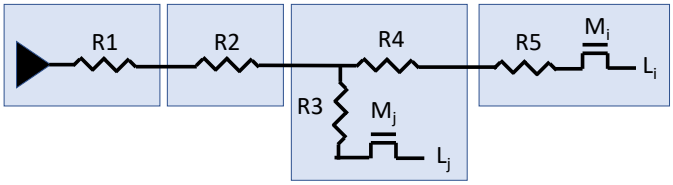

Fig. 4. The portion of a tree driving leaves L_i and L_j in a polyomino P of four tiles. The common path resistance $R(P, M_i, L_j) = R(P, M_j, L_i) = R1 + R2$.

A. Models

We consider the following alternative models for estimating $D(P, L, F, C)$ for a given tree T. The models contain parameters B and K having various dependencies. The B parameters are "baseline" values independent of fanout. The K parameters reflect the fanout- or configuration-dependent portion of the delay.

$$B(P, L, F) \tag{1}$$

$$B(P, L, F) + K(F) \sum_M X(M) \tag{2}$$

$$B(P, L, F) + \sum_M K(M, F)\, X(M) \tag{3}$$

$$B(P, L, F) + \sum_M K(M, L, F)\, X(M) \tag{4}$$

$$B(P, L, F) + K(F) \sum_M R(P, M, L)\, X(M) \tag{5}$$

$$B(P, L, F) + \sum_M K(M, F)\, R(P, M, L)\, X(M) \tag{6}$$

$$B(P, L, F) + \sum_M K(M, L, F)\, R(P, M, L)\, X(M) \tag{7}$$

(All parameters also have an implied dependence on the tree T, not shown above.)

Equation (1) is a fanout-independent model. In all other cases the summation is over those muxes M which need not be active to reach L but would (if active) add incremental capacitive loading on the path to L.

Equations (2), (3) and (4) add fanout dependence in increasingly expressive ways. Equation (2) depends on the number of active fanouts, while equations (3) and (4) depend on the particular set of active fanouts. In equation (3), the impact of activating mux M is independent of L, the leaf whose delay we are estimating; in equation (4), the impact may differ for each L.

Equations (5)-(7) are similar to equations (2)-(4), respectively, but include the novel common path resistance term that is a key contribution of this paper. The inspiration for introducing common path resistance into the models in this way is the role it plays in the Elmore delay formula. In the Elmore formula, the contribution to the delay at a leaf L from a capacitive load C is proportional to RC, where R is the resistance of the common portion of the paths to L and C. Observe that in equations (5)-(7), $K \bullet X$ plays the role of the capacitive load downstream of mux M.

Note that our $B(P, L, F)$ parameters depend on both the polyomino P (reflecting any layout irregularities) and leaf L (the leaf for which we are estimating delay). This differs from most previous work (such as [9]). So we also consider restrictions of the above models which split these impacts into two separate additive terms:

$$B(P, L, F) = B_L(L, F) + B_P(P, F) \tag{8}$$

We also consider simplifications of the above models that eliminate the dependence of the K parameters on the transition, F.

B. Data

Transistor-level netlists were extracted for each tile type of a prototype commercial FPGA architecture targeting an

advanced CMOS process representative of the challenges explained in Section 1.

We selected ten example delay trees, covering inter-tile segments of every length present in the architecture:

- An L1 segment driven by a mux in one tile that drives muxes in the same tile and one of the four immediately adjacent tiles.
- An L2 segment that drives muxes in the tiles at distance 1 and 2, either vertically or horizontally.
- An L4 segment that drives muxes 4 tiles away, either vertically or horizontally.
- An LL segment that drives muxes a long distance away, either vertically or horizontally.

The L2 segments are especially interesting because they have fanouts at an intermediate position (distance 1), and stretching can occur either upstream or downstream of these.

For a tree having N possible fanouts, we further selected: all configurations with one or two active fanouts; up to 10 configurations with each number of active fanouts from three through $N-1$; and the one configuration with all N fanouts active. This gives us a set of $[T, C]$ pairs.

For each such pair, we selected a polyomino from the set of applicable polyominos in a randomized round-robin way so every possible polyomino is used at least once. The set of applicable polyominos provide considerable variation in physical segment length due to array irregularities. The polyominos also vary the position along the segment at which stretching occurs, relative to the driving and receiving tiles. This gives us a set of $[T, P, C]$ triples.

For each such triple, we performed a SPICE simulation to get the delay to each leaf active under the configuration C for both rising and falling transitions. The same upstage driver was used for all triples involving the same tree to control for any dependence on the transition time of the upstream stage.

Overall, the data includes a total of 43,000 SPICE delay measurements involving 1,731 distinct polyominos.

C. Fitting Procedure

The typical way to determine parameters for a linear model is to use least-squares regression. However, in the present case, where we expect errors in the delays from SPICE (i.e., differences between SPICE and silicon) to be relatively small compared to model fitting errors, it is more appropriate to minimize the worst-case absolute value of the errors [13]. This quantity, also known as the L_∞ norm, can be readily minimized by linear programming. It also may be more indicative of guarantees we wish to make to users about their applications' worst-case timing. Finally, this process also yields parameters that are less sensitive to the exact distribution of training data (i.e., overrepresentation of one type of data point versus others).

A much smaller cost is also imposed on the sum of the absolute value of the errors to break ties, and also on the sum of the K parameters to reduce overfitting.

D. Evaluation Method

We first evaluate the expressiveness of each model by doing a fit to all the data and reporting the number of K parameters employed and the maximum error obtained. (We focus on the K parameters because they seem to be more prone to overfitting.)

Then we did 30 trials of cross validation. Training sets were selected as follows. We begin by identifying all the $[T, P, C]$ triples for which delays are available. Then for each trial, we select a randomized subset of these triples subject to the constraint that it provides sufficient delay data to determine all parameters. For example, for each $[T, P, L, F]$ tuple, the training set must include at least one tuple $[T, P, C]$ where C activates L. Otherwise, $B(P, L, F)$ will be indeterminate. (For further details, see the Appendix.) We make no claim that this method of selecting training sets is especially good. It is merely a reasonable method we can use to evaluate the relative predictiveness of the models.

The resulting training sets each cover approximately ¾ of the delay values. We train each model using the delay data covered by the training set, and predict the remaining ¼ of the delay values.

IV. Results

We observed that eliminating the dependence of the K parameters on the transition F (fall/rise) did not significantly degrade either the expressiveness or cross validation results. From here on, we assume that dependence is eliminated, halving the number of K parameters.

Table I shows the results. The metrics have these meanings:

- *FittingErr*: the maximum absolute value of the errors when fitting the model to all available data.
- *AvgErr*: the average absolute value of the error on the validation set.
- *AvgRelErr*: the average of the absolute value of the error divided by the correct (SPICE) delay on the validation set, expressed as a percentage.
- *RMSErr*: the root-mean-square error on the validation set.
- *MinErr*, *MaxErr*: the range of errors on the validation set.

All metrics (except for AvgRelErr) are normalized so the worst FittingErr is 25.

We use AvgErr as the primary means of ranking the models; the other metrics are just for further information.

The results indicate that the model of row 13 is the most accurate. It uses the novel common path resistance terms and merged B parameters. AvgErr is reduced by 46% and MaxErr by 73% compared to the best model without these techniques (in row 2). We attempt to elucidate why by examining the data more closely.

Rows 1 and 8 are fanout-independent models. These have the worst errors, both in fitting and cross validation. The other rows show fanout-dependent models of increasing complexity. These have parameters K dependent on: only the tree (10 constants K, one per tree); which mux is adding loading (114 parameters $K(M)$); or on both the mux and which leaf's delay is being estimated (1336 parameters $K(M, L)$).

TABLE I. RESULTS

Row	Type of B Params	Includes Common Path R?	Number of K Params	Model	Fitting Err	Cross Validation				
						Avg Err	Avg Rel Err (%)	RMS Err	Min Err	Max Err
1	split	no	0	$B_L(L,F) + B_P(P,F)$	25.00	14.03	16.3	16.9	-44.4	15.7
2	split	no	10	$B_L(L,F) + B_P(P,F) + K \Sigma X(M)$	18.00	3.62	4.8	6.4	-22.9	35.4
3	split	no	114	$B_L(L,F) + B_P(P,F) + \Sigma K(M)X(M)$	16.78	3.91	5.2	6.8	-24.5	38.2
4	split	no	1336	$B_L(L,F) + B_P(P,F) + \Sigma K(M,L)X(M)$	12.12	12.53	14.8	16.8	-38.3	78.3
5	split	yes	10	$B_L(L,F) + B_P(P,F) + K \Sigma R(P,M,L)X(M)$	12.90	3.62	4.7	6.5	-20.7	27.4
6	split	yes	114	$B_L(L,F) + B_P(P,F) + \Sigma K(M)R(P,M,L)X(M)$	9.24	4.89	6.3	10.0	-32.0	52.8
7	split	yes	1336	$B_L(L,F) + B_P(P,F) + \Sigma K(M,L)R(P,M,L)X(M)$	8.67	7.53	9.0	15.6	-35.8	105.7
8	merged	no	0	$B(P,L,F)$	23.57	13.69	15.7	16.6	-48.2	9.0
9	merged	no	10	$B(P,L,F) + K \Sigma X(M)$	11.37	5.25	6.4	6.7	-18.1	31.8
10	merged	no	114	$B(P,L,F) + \Sigma K(M)X(M)$	10.16	3.09	4.2	5.4	-19.4	29.1
11	merged	no	1336	$B(P,L,F) + \Sigma K(M,L)X(M)$	6.65	5.75	6.5	7.6	-29.1	21.5
12	merged	yes	10	$B(P,L,F) + K \Sigma R(P,M,L)X(M)$	9.20	6.58	8.4	8.3	-23.9	3.3
13	merged	yes	114	$B(P,L,F) + \Sigma K(M)R(P,M,L)X(M)$	5.38	1.95	2.8	3.2	-12.6	9.4
14	merged	yes	1336	$B(P,L,F) + \Sigma K(M,L)R(P,M,L)X(M)$	3.79	1.99	2.8	3.4	-16.4	24.9

Looking at rows 1-4 (for split B) or 8-11 (for merged B) we see a clear trend that the more K parameters, the lower the training error (FittingErr). This confirms the importance of configuration dependence. However, by the time we reach 1336 parameters, the cross-validated error (AvgErr) goes back up, evidence that overfitting has started to occur.

This is where using common path resistance can help. Comparing row 13 to row 11, we see that adding common path resistance enables us to reduce the number of K variables by more than 10x while avoiding overfitting and reducing errors.

Now we turn to the benefits of the merged B parameters. Comparing row 1 to row 8, we see that for fanout-independent models the conventional split B parameters do nearly as well. However, for higher-accuracy fanout-dependent models with 114 or more K parameters, the use of the merged B parameter is indeed beneficial. This can be seen by comparing row 13 to row 6, or row 10 to row 3. We surmise that if the model lacks K parameters and their ability to accurately capture fanout-dependence, the power of the merged B parameters is misused to overfit the fanout dependence.

Unfortunately, it is difficult to compare our results directly to previous commercial FPGA delay models since those are proprietary. We believe they are most similar to the models of rows 2 and 3. An open-source model for a commercial FPGA is described in [9], but it is just a lightweight model intended for use in timing-driven place and route. It is fanout-independent and has a separate additive term for floorplan irregularities, similar to our row 1. It was evaluated only against estimates from the proprietary FPGA software, which in turn are an approximation to SPICE.

V. CONCLUSIONS

Accurate estimates for FPGA routing delay are essential for a high-quality design flow. However, estimating delays for inter-tile routing in FPGAs is particularly difficult due to the combined effects of fanout-dependence and floorplan irregularities. We proposed and evaluated a number of regression models to predict such delays as a function of the routing tree topology, the set of active fanouts (or configuration), and the relevant layout tiles. We showed:

- Models must depend on the configuration in order to achieve acceptable accuracy.
- For the best accuracy, models must account for interactions between the configuration and floorplan irregularities. Incorporation of common path resistance into a model is a novel and effective way to accomplish

this without unduly increasing model complexity. In particular, it is not necessary to have any parameter (which must be determined from SPICE data) depend on both the configuration and the floorplan irregularities. This helps limit the amount of SPICE data necessary to determine the model parameters.

- With the more accurate configuration-dependent models, it is beneficial to allow for layout irregularities having different impacts at different fanouts. We showed this can be done effectively using our merged B parameters.

APPENDIX: TRAINING SET SELECTION ALGORITHM

In the explanation below, $L(M)$ refers to the leaf driven directly by mux M.

We pick a training set for each tree T as follows. Begin by sorting the list of all $[T, P, C]$ triples for which we have SPICE data into a random order. Then pass through the list, selecting a triple for inclusion in the training set if and only if it meets any of these criteria:

- $\exists M \in C$ such that we have not yet selected a triple $[T, P, C']$ where $M \in C'$ (Needed to estimate $B(P, L(M), F)$.)
- $\exists M_1, M_2 \in C$ such that we have not yet selected a triple $[T, P', C']$ where $M_1, M_2 \in C'$. (Needed to estimate impact of turning on one mux on the delay at the leaf driven by the other, and hence estimate $K(M_1, L(M_2))$ and $K(M_2, L(M_1))$.
- $\|C\| > 1$ and $\exists M_1 \in C$ and $M_2 \notin C$ such that we have not yet selected a triple $[T, P', C']$ where $\|C'\| > 1$ and $\|C' \cap \{M_1, M_2\}\| = 1$. (Needed to distinguish impact of turning on M_1 versus M_2 on delay at some other leaf, and hence assign unique values to $K(M_1)$ and $K(M_2)$.)
- $\exists M \notin C$ such that we have not yet selected a triple $[T, P', C']$ where $M \notin C'$. (Needed to estimate impact of not turning on M on the delay to some other leaf, and hence estimate $K(M)$.)

ACKNOWLEDGMENTS

The authors thank Nizar Abdallah, Hassan Hassan and Volker Hecht for their suggestions. Matthew Kelly and Tony Liao contributed to the infrastructure used in this project.

REFERENCES

[1] W. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," Journal of Applied Physics, 19(1), pp. 55-63, 1948. https://doi.org/10.1063/1.1697872

[2] V. Betz, J. Rose, and A. Marquardt, Architecture and CAD for Deep-submicron FPGAs. Norwell, MA: Kluwer Academic Publishers, 1999.

[3] L. Pillage and R. Rohrer, "Asymptotic waveform evaluation for timing analysis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 9(4), pp. 352-366, 1990. https://doi.org/10.1109/43.45867

[4] D. Lewis, V. Betz, D. Jefferson, A. Lee, C. Lane, P. Leventis, S. Marquardt, C. McClintock, B. Pedersen, G. Powell, S. Reddy, C. Wysocki, R. Cliff, and J. Rose, "The Stratix routing and logic architecture," Proceedings of the ACM/SIGDA Eleventh International Symposium on Field Programmable Gate Arrays (FPGA), pp. 12-20, 2003. https://doi.org/10.1145/611817.611821

[5] C. Chiasson and V. Betz, "COFFE: fully-automated transistor sizing for FPGAs," International Conference on Field-Programmable Technology (FPT), pp. 34-41, 2013. https://doi.org/10.1109/FPT.2013.6718327

[6] T. Martin, G. Gréwal, and S. Areibi, "A machine learning approach to predict timing delays during FPGA placement," IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 124-127, 2021. https://doi.org/10.1109/IPDPSW52791.2021.00026

[7] M. Raman, N. Abdallah, and J. Dunoyer, "An artificial intelligence approach to EDA software testing: application to net delay algorithms in FPGAs," 20th International Symposium on Quality Electronic Design (ISQED), pp. 311-316, 2019. https://doi.org/10.1109/ISQED.2019.8697652

[8] M. Guillorn, et al., "FinFET performance advantage at 22nm: an AC perspective," IEEE Symposium on VLSI Technology, pp. 12-13, 2008. https://doi.org/10.1109/VLSIT.2008.4588544

[9] P. Maidee, C. Neely, A. Kaviani and C. Lavin, "An open-source lightweight timing model for RapidWright," International Conference on Field-Programmable Technology (FPT), pp. 171-178, 2019. https://doi.org/10.1109/ICFPT47387.2019.00028

[10] C. Chiasson and V. Betz, "Should FPGAs abandon the pass-gate?," 23rd International Conference on Field programmable Logic and Applications (FPL), pp. 1-8, 2013. https://doi.org/10.1109/FPL.2013.6645511

[11] J. Greene, S. Kaptanoglu, W. Feng, V. Hecht, J. Landry, F. Li, A. Krouglyanskiy, M. Morosan and V. Pevzner, "A 65nm flash-based FPGA fabric optimized for low cost and power," Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA), pp. 87-96, 2001. https://doi.org/10.1145/1950413.1950434

[12] J. Greene, E. Hamdy and S. Beal, "Antifuse field programmable gate arrays," Proceedings of the IEEE, 81(7), pp. 1042-1056, 1993. https://doi.org/10.1109/5.231343

[13] R. Shrager, E. Hill, "Nonlinear curve-fitting in the L_1 and L_∞ norms," Mathematics of Computation, 34(150), pp. 529-541, 1980. https://doi.org/10.1090/S0025-5718-1980-0559201-X