

Performance Analysis by Topology Indexed Lookup Tables

Patrika Agarwal^{1,2} Arvind Vidyarthi^{1,3} Patrick H. Madden²
Sun Microsystems¹ SUNY Binghamton CSD² and ECE³

Abstract—Accurate analysis of VLSI interconnects is essential to the performance-driven synthesis and layout of integrated circuits. Existing techniques are based on either simulation, analytic formulas, or small-scale table lookup. There are tradeoffs in compute time and accuracy.

In this paper, we present an effective approach which captures the accuracy of SPICE while remaining close to Elmore delay in terms of computational overhead. Our method is based on topology indexed lookup tables (TILT), and uses a extremely large database of precomputed values.

I. INTRODUCTION

Interconnect has become the dominating factor in determining circuit performance in deep submicron designs[3]. During the early stages of design, simple estimates or delay bounds are used to guide optimization. In the later stages of physical design, however, accuracy is required. Unfortunately, more accurate delay models can have a heavy price in terms of computing overhead.

There are opportunities for design optimization when accurate analysis is used. In one recent work[13], gate sizing optimization was interleaved with placement and routing. By interleaving these steps, circuit speed was improved while at the same time, power consumptions were reduced. In [10], Elmore delay[4] was used to guide placement—while far from accurate, Elmore delay provided an improvement over simple wire length objectives.

Further improvements in performance are possible if the sizing and physical design steps can be more tightly integrated. Capitalizing fully on this, however, requires an extremely fast and accurate method for delay and power analysis. Within the “inner loop” of a physical design tool, performance may need to be calculated hundreds of thousands of times; accurate circuit simulation methods such as SPICE are clearly too slow.

Our work in this paper is to present a method that in some sense gives the “best of both worlds.” We develop a delay estimation approach that gives SPICE-like accuracy, with Elmore-like run times. On experiments with large numbers of randomly generated nets, we obtain delay estimate errors that are normally less than 5%, and for many nets, less than 1%. The compute time to obtain these delay estimates is comparable to that of Elmore delay computation; certainly fast enough to be used within physical design tools.

We refer to our approach as “topology indexed lookup tables (TILT).” While lookup tables for delay and power analysis are not new, we utilize tables that are extraordinarily large. We precompute delay values for *hundreds of thousands or*

millions of interconnect trees, and then use fast hashing-based lookup to obtain delay values during physical design.

Two trends have made our approach practical. First, there has been a dramatic increase in memory capacity in typical workstations—this allows the creation of large in-memory databases. Recently, [2] used this fact to develop a method for fast interconnect tree length estimation. A second trend is in the number of sinks on a typical interconnect net: lower drive strengths and comparatively increased wiring capacitance has motivated the decomposition of high degree nets into buffered tree of small degree nets. Buffering is extensive[8], and most interconnect trees have only a few sinks. We also note that the connecting wires are also relatively short; delay in unbuffered wires increases exponentially with length.

We stress that the general principle is not new: table lookup has a long history in computer aided design, and many other fields. What is new is the size of the database, allowing us to perform analysis for the vast majority of interconnect trees without incurring a prohibitively high compute overhead. While we cannot perform delay analysis for *all* possible nets, we can capture an interesting, useful, and relatively large subset.

Due to space constraints, our focus in this paper is on delay estimation on RC trees. The method we present can easily include input slope information and inductance effects in the “input” to our lookup table, and can record output slope, switching power, and a number of other effects in the “database.” We can also compute the delay and slope *to each sink* in an interconnect tree. We are integrating the method presented into the *feng shui 5.0* placement tool developed by our group[7]. The speed of analysis, coupled with sink-specific performance information, is a key element of our overall approach to the integration of gate sizing and buffer insertion with placement.

II. PREVIOUS WORK

The industry standard for *accuracy* is the SPICE simulation method; initially developed at UC Berkeley, versions of the tool are available from a number of commercial vendors, and many variants have been developed to offer different types of analysis. The approach relies heavily on circuit simulation; because of this, it is also exceptionally slow, and is unsuitable for use during physical design.

To allow for both acceptable compute times, and reasonable accuracy, a great deal of research has gone into analytic methods. The major focus has been on estimating delay in RC

networks, assuming that the drive transistors can be modeled as a resistor.

In 1948, Elmore[4] introduced a general approach for calculating the propagation delay of a linear system given its transfer function. If $h(t)$ is the impulse response at a node of an RC tree, the transfer function $H(s)$ of the circuit, which is the Laplace transform of $h(t)$, and can be represented as:

$$H(s) = \int_0^{\infty} h(t)e^{-st} dt = \sum_{k=0}^{\infty} \frac{(-1)^k}{k!} s^k \int_0^{\infty} t^k h(t) dt$$

The k th-circuit response moment, m_k is defined as:

$$m_k = \frac{(-1)^k}{k!} \int_0^{\infty} t^k h(t) dt$$

These time moments provide excellent measures of delay, rise times[9], and so on.

While ‘‘Elmore delay’’ was not particularly well known for many years, Penfield and Rubinstein[11] used the formulation to analyze RC trees; following this, the metric rapidly increased in popularity. Elmore Delay is the first moment $m_1 = \int_0^{\infty} t \cdot h(t) dt$ of the impulse response. Additional moments m_2, m_3, \dots can be used to derive more accuracy.

Elmore Delay for a RC tree with N nodes at the i_{th} node can be computed as:

$$\tau_i = \sum_{k=1}^N R_{ik} C_k$$

where R_{ik} be the total resistance of the portion of the unique path from the source node to the i_{th} node where $0 < i \leq N$,

$$R_{ik} = \sum R_j \Rightarrow (R_j \in [path(i \rightarrow s) \cap path(i \rightarrow s)])$$

This is simple to calculate, and models interconnect delay much more accurately than simple lumped RC metrics.

A number of authors have subsequently adapted or improved the Elmore delay model. For example, Alpert[1] introduced a ‘‘k-factor’’ into the common Elmore Delay calculation, thereby taking into consideration the resistive shielding of downstream capacitance. A variety of ways of developing this ‘‘k-factor’’ were presented, with varying impact. While delay estimates were dramatically improved, the approach becomes computationally expensive as the number of RC elements increases.

In general, Elmore delay offers good quality results with very modest compute times. There are, however, a number of drawbacks of Elmore Delay. First, it is a first order approximation, and cannot handle inductance effects[5], [9]. Second, it cannot be applied to estimate the delay for interconnect lines with ramp input sources[6]. Third, it also ignores resistive shielding of downstream capacitance[1]. All of these effects can be handled by the method we present in this paper.

As a more accurate alternative to Elmore delay, Pileggi and Rohrer[9] introduced an efficient AWE (asymptotic waveform evaluation) approach for linear RLC interconnect models. They showed that higher orders of approximation are obviously desirable for improving the accuracy of the response approximation and also for general handling of nonmonotone

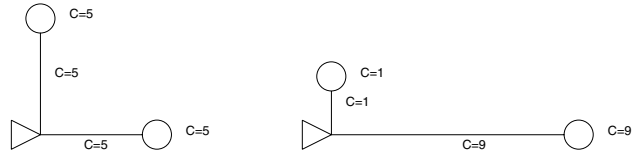


Fig. 1. Nets with differing topologies, but identical performance under traditional table lookup methods.

response waveforms arising from circuits which contains multiple input signals, nonequilibrium initial conditions, floating capacitors, complex poles etc. AWE computation requires more time than Elmore delay, and this makes the approach less attractive for physical design applications.

Delay estimation research has focused on improving compute times while also increasing accuracy. In general, the physical modeling of the circuits is quite simplified. The transistors of logic elements are frequently modeled as resistors, and the effects of fast and slow input transition times are ignored. As device and interconnect sizes scale, the impact of inductance, overshoot and ringing, non-linear transistor behavior, and so on, become more pronounced.

The method we employ here – table lookup – is also common. Synopsys PowerARC, for example, uses relatively small tables indexed by net capacitance and input slope. Each table has a handful of values (usually on the order of 50 or less); a simple interpolation scheme is used. A shortcoming of this approach, which we address here, is that *interconnect topology is ignored*. Two nets with identical capacitances and input slopes will have identical delays using current table lookup methods—but in practice, the delay to any given sink depends a great deal on topology. Figure 1 illustrates this point.

III. TOPOLOGY INDEXED LOOKUP TABLES

As was mentioned in the introduction, our approach to delay estimation is through table lookup. Our motivation is to have a method that is both extremely fast and extremely accurate, for use in our current placement, routing, and gate sizing work. Our method relies on a database of precomputed results; we make queries into this database during physical design. We use and store *entire trees* in our database; tree matching is used heavily in our approach.

In this section, we first discuss tree patterns and the creation of our database, and then show how we query for individual trees.

A. Tree Patterns

The number of possible patterns for a net with given number of nodes is constant. For example, a net with a single source and single sink can have only one topology: a wire connecting the source to the sink. If there are three nodes, we have only two possible patterns. Four node trees have four patterns, and so on; this is illustrated in Figure 2. While the number of topologies for a tree with n nodes is exponential, practical values of n are quite small—buffering[8] is used to break up nets with large (logical) fan-out.

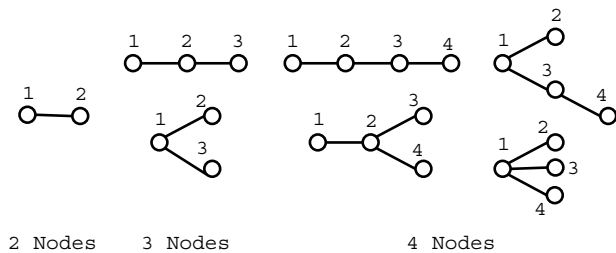


Fig. 2. Possible tree patterns for small nets.

For each net, we have only a handful of possible topologies. For each edge in a net, the edges could be of arbitrary length—but these too are constrained. Low drive strengths of most transistors results in wire lengths being short; again, buffering is used heavily. For database construction, we restrict each edge in each pattern to a relatively small set of possible lengths, resistance per unit length, and capacitance per unit length. For each node in a tree, we also restrict the number of possible drive strengths or sink capacitances. Our database is populated with a large number of possible tree patterns, using edges of varying lengths.

The number of lengths considered for each edge impacts the database size considerably; the number of database entries for a pattern with e edges and l possible lengths is obviously l^e . The number of lengths, and their range, must be chosen with care; in our experiments, we have designed the database to fit easily into the memory of an ordinary workstation; as memory prices decrease, increasing the number of edge lengths will increase accuracy.

In this paper, we consider edge lengths to be multiples of a “step size.” Other strategies are possible, and we are currently investigating methods to skew the distribution of edge lengths to more closely follow the lengths observed in real circuit designs.

B. Database Creation

Our approach to database construction is relatively simple. First, for different numbers of nodes and all possible patterns of trees, a database without the delay values is created. We then populate the database with delay values by constructing a SPICE deck (using a distributed model for better accuracy) for each tree, and then actually running SPICE. We use driver resistance, interconnect resistances and capacitances, and sink capacitances suggested by ITRS[12]. We measure the 50% delay value, and store this along with the other tree parameters in the database. We could clearly also store the rise and fall transition times, as well as switching power.

This process can be time consuming as the number of trees in the database increases. Fortunately, this needs to be done just once for the database creation, and can obviously be distributed easily across a number of workstations.

C. Tree Matching

During physical design, our tools repeatedly need to determine the delay performance of interconnect nets—for each

tree, we query the database. Our database contains only trees of relatively limited size (in terms of the number of nodes); for large fan-out nets, we fall back to Elmore delay. For trees with a supported number of nodes, we perform “tree matching.”

In most cases, an exact match of a database entry is not possible, so we use a simple interpolation method. We construct upper and lower bound trees by modifying the query tree. We simply round each edge “up” or “down” to a length that is considered in our database. This is illustrated in Figure 3. The upper and lower bound trees have exact matches in the database, and we interpolate the delay values for the two bound trees to obtain a delay estimate for the query tree.

As finding a match in the database is a key operation in our approach, we have taken care to ensure that this is efficient. We currently perform the match by first converting a query tree into a canonical ASCII string. We then use this string with the GNU hash functions (from the GNOME library); these routines are highly optimized, allowing quick retrieval of matching trees.

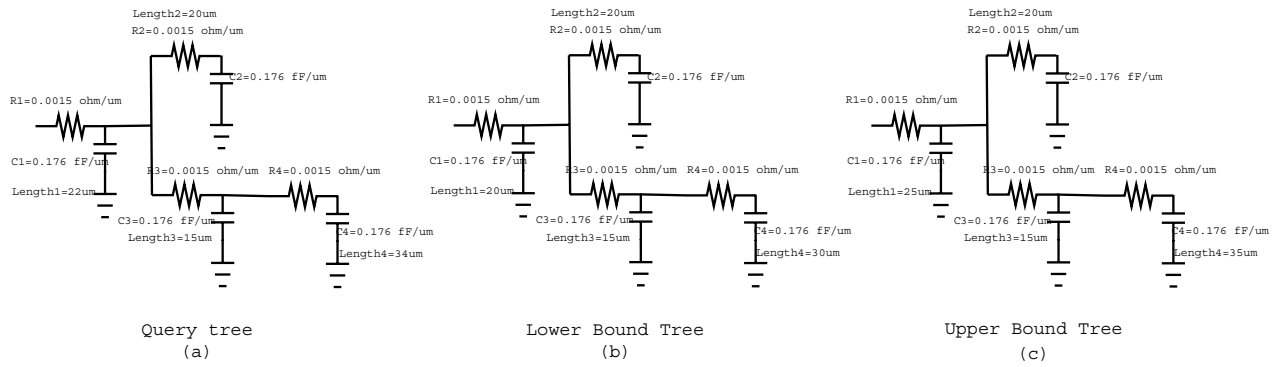
IV. EXPERIMENTAL RESULTS

In our experiments, we were interested in how closely we could match the results of SPICE, using databases of reasonable size. Table I shows the minimum, maximum, and average error in delay with respect to SPICE, for both the Elmore delay, and using our table lookup method. The edges in our database were of restricted lengths; the allowed lengths are determined by “step size,” and the total number of trees in the database are also given.

In these experiments our database contains trees with driver resistances of 20, 40, and 60 Ω , sink capacitances of 0pf or 2pf, interconnect resistance of 0.15 Ω per micron, and interconnect capacitance of 0.176fF per micron. Edge lengths ranged from 10 to 50 microns (inclusive); for trees with two nodes, the database step size was a one micron, while higher fan-out nets had larger step sizes. Our parameters are based on 0.18 μ technology. Query trees used to test accuracy have between two and six nodes, randomly chosen topologies, randomly chosen drive strengths and sink capacitances, and randomly chosen edge lengths from 10 to 50 microns in one micron steps. We limit our consideration to RC trees; in experiments with RLC trees, results are similar.

For trees with two, three, four, and five nodes, our average error ranges from 0% to just over 3%, while Elmore delay has a much wider range. This should not be surprising: we have large numbers of trees in the database, and while an exact match is rare, the interpolation between two trees produces a delay estimate that is very close. The table lists the number of edges that match in any given query (and if there is more than a single edge match).

For six nodes, we increased the step size, significantly reducing the number of trees in the database, and also increasing the error. There is a tradeoff of accuracy for database size; as available memory increases, accuracy will improve. Further, we are developing a new interpolation scheme that can significantly reduce database sizes without impacting accuracy.



For step size of length in Database = 5um

Fig. 3. Approximate matching of trees; delay values of the query tree are obtained by interpolating values extracted from the database.

Nodes	Trees	Stepsize	Edge Matches					Percentage error relative to SPICE						Runtime		
			1	2	3	4	5	Elmore			TILT			SPICE	Elmore	TILT
							Min	Max	Ave	Min	Max	Ave	ms	ms	ms	
2	246	1	1000	-	-	-	0.45	44.26	29.36	0.00	0.00	0.00	21426	27	22	
3	5292	2	469	268	-	-	0.01	50.01	23.80	0.00	3.34	0.16	24257	38	51	
4	47916	4	434	158	18	-	0.04	47.55	21.95	0.00	8.80	1.09	27762	49	77	
5	124416	8	370	107	15	0	0.01	46.20	21.57	0.00	15.93	3.28	31656	60	99	
6	64152	20	287	28	3	0	0.01	66.84	22.82	0.00	71.20	11.36	34395	72	116	

TABLE I

PERCENTAGE ERROR RELATIVE TO SPICE FOR TREES OF VARIOUS SIZES. COMPUTE TIMES OF OUR METHOD AND ELMORE DELAY ARE COMPARABLE; BOTH ARE ORDERS OF MAGNITUDE FASTER THAN SPICE.

V. CONCLUSION

In this paper, we have presented a “topology indexed lookup table” approach to delay analysis. The key insight is that with the large amount of memory available on modern workstations, this approach can quickly provide accurate results. There is a clear trade-off of memory (which is relatively abundant) for compute time (which is in short supply). While we focus on delay analysis here, consideration of other performance metrics, and the use of complex models, is easily supported.

Modern VLSI circuits commonly have very low fan-out nets; the input capacitance of down-stream gates makes high fan-out interconnect very unattractive. Most tree edges are also short: longer edges are normally buffered to provide delay reduction and noise immunity. Because of this, it is possible and practical to develop a database of pre-computed interconnect trees which can then be used to obtain delay estimates. Our current database holds several hundred thousand trees—larger databases, supporting a wider range of driver strengths and sink capacitances, are clearly practical.

We are currently working on an improved interpolation method, that offers better accuracy with much smaller tables. Details of our integration of this delay analysis method with our current physical design work will be presented in a subsequent paper.

REFERENCES

[1] C. J. Alpert, A. Devgan, and C. V. Kashyap. RC delay metrics for performance optimization. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 20(5), 2001.

[2] C. Chu. FLUTE: fast lookup table based wirelength estimation technique. In *Proc. Int. Conf. on Computer Aided Design*, pages 696–701, 2004.

[3] J. Cong, L. He, C.-K. Koh, and P. H. Madden. Performance optimization of VLSI interconnect layout. *Integration, the VLSI Journal*, 21:1–94, 1996.

[4] W. C. Elmore. The transient response of damped linear networks with particular. *Journal of Applied Physics*, 19(1):55–63, January 1948.

[5] Y. I. Ismail, Eby G. Friedman, and Jose L. Neves. Equivalent elmore delay for rlc trees. In *Proc. Design Automation Conf*, 1999.

[6] A. B. Kahng, K. Masuko, and S. Muddu. Analytical delay models for interconnects under ramp input. Technical Report TR-950032, UCLA, September 1995.

[7] A. Khatkhate, C. Li, A. R. Agnihotri, M. C. Yildiz, S. Ono, C.-K. Koh, and P. H. Madden. Recursive bisection based mixed block placement. In *Proc. Int. Symp. on Physical Design*, pages 84–89, 2004.

[8] P. Osler, L. Scheffer, P. Saxena, and D. Sylvester. The great interconnect buffering debate: Are you a chicken or an ostrich? In *Proc. Int. Symp. on Physical Design*, page 61, 2004.

[9] L. T. Pillage and R. A. Rohrer. Asymptotic waveform evaluation for timing analysis. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 9(4):352–366, April 1990.

[10] H. Ren, D. Z. Pan, and D. S. Kung. Sensitivity guided net weighting for placement driven synthesis. In *Proc. Int. Symp. on Physical Design*, pages 10–17, 2004.

[11] J. Rubinstein, P. Penfield, Jr., and M. A. Horowitz. Signal delay in RC tree networks. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, CAD-2(3):202–211, July 1983.

[12] Semiconductor Industry Association. *International Technology Roadmap for Semiconductors*, 2001.

[13] M. Vujkovic, D. Wadkins, B. Swartz, and C. Sechen. Efficient timing closure without timing driven placement and routing. In *Proc. Design Automation Conf*, pages 268–273, 2004.