Efficient Time-Domain Simulation of Frequency-Dependent Elements

Sharad Kapur David E. Long Jaijeet Roychowdhury
Lucent Technologies Bell Laboratories

Abstract

We describe an efficient algorithm for time-domain simulation of elements described by causal impulse responses. The computational bottleneck in the simulation of such elements is the need to compute convolutions at each time point. Hence, direct approaches for the simulation of such elements require time $O(N^2)$, where N is the length of the simulation. We apply ideas from approximation theory to reduce this complexity to $O(N \log N)$ while maintaining double-precision accuracy. The only restriction imposed by our method is that the impulse response $h(t)$ gets "smoother" as t goes to infinity. Essentially all physically reasonable impulse responses have this characteristic. The ideas presented can also be applied to time-domain simulation of elements described in the frequency domain, including those characterized by measured data. In this paper, we demonstrate the efficiency of the algorithm by applying it to the simulation of lossy transmission lines.

1 Introduction

Recently, increasing speeds have made it critical that designers of digital systems consider phenomena that were traditionally viewed as "analog." With bit rates in high-speed digital systems (e.g., ATM switches) approaching one gigabit per second, the treatment of distributed effects is now necessary in most state-of-the-art designs. For example, transmission lines are often used to represent interconnections in high-speed systems. Distributed effects are modeled at differing levels of sophistication, ranging from ideal lines to elements incorporating loss and skin effect. Often, analytical models prove to be inadequate for capturing the electrical behavior of involved geometries and one must resort to measured data (e.g., s-parameters over a frequency range) for accurate characterization. Simulating such models in the frequency-domain is relatively straightforward using techniques such as harmonic balance. However, for digital systems, transient analysis is often more natural. In this paper, we describe a general and efficient method for the time-domain simulation of elements such as transmission lines and resistors with skin-effect.

The problem of simulating distributed effects in digital circuits has received widespread attention over the last few years. One common approach is to use a lumped-element approximation [5]. This method has the advantages of being simple and working easily with existing simulators. However, there is a trade-off between accuracy and efficiency. With few lumps, the simulation will exhibit artifacts that would not be present if the distributed element were used. Increasing the number of lumps to the point where the accuracy is good often leads to long simulation times. More robust methods involve computing convolutions with the distributed element's time-domain impulse response at each point in the transient analysis. For efficiency, they either enforce or assume certain forms for the response. One such approach is based on reduced-order models. In the most popular schemes, the frequency-domain response is matched by a Padé approximation, which can then be put into a partial fraction form [3, 8, 9]. When this form is transformed to time domain, the result is a sum of exponentials. The latter is suitable for a recursive convolution computation, so only a constant amount of work is required at each transient time point. One potential problem is that the computed Padé approximation may have poles with positive real parts, even when the circuit is in fact stable. It is also possible to directly approximate the time-domain response, again obtaining a form that is suitable for recursive convolution [11]. Other methods, such as that described by Roychowdhury *et al.* [10] for simulating transmission lines, rely on the particular impulse responses for certain types of elements.

Our approach is also based on computing convolutions, but we make only one weak assumption about the time-domain response $h(t)$: that it becomes "smoother" as t goes to infinity. This is true of essentially all physically reasonable impulse responses. The inspiration behind our method comes from some of the recent work on fast n-body particle simulation algorithms [7]. These methods use the fact that the influence of a group of particles in a region of space

that is far from the group is slowly varying and hence can be accurately approximated by a low-order polynomial. If there are a large number of particles in the group, then the influence can be evaluated much faster using this approximation than by a direct scheme. This observation can be exploited to yield a divide-and-conquer algorithm that evaluates the influence on each of the n particles in $O(n)$ or $O(n \log n)$ time, compared with $O(n^2)$ time for the direct method. Similar ideas have been used in a number of other areas [2, 6]. We can apply these techniques in the context of circuit simulation as follows. For an impulse response that gets smoother as time increases, the convolution with the tail of the response varies slowly. Hence, we can approximate this part of the convolution with a low-order polynomial. At each time point, most of the convolution can be computed by simply evaluating this polynomial.

2 Time-Domain Simulation and Convolutions

Circuit elements with linear frequency dependencies can be easily described in the frequency domain by the transfer function $H(s)$ which gives the input-output relationship of the element. For an input $X(s)$, the output is simply $Y(s) = H(s)X(s)$. In a time-domain simulation of such an element, the multiplication becomes a convolution.

$$y(t) = \int_0^t h(t-\tau)x(\tau)\,d\tau \tag{1}$$

At each simulation time point, we need to compute this integral. Numerically, each integral will involve a sum over all previous time points. For simplicity, we assume that the above integral is computed using the rectangular rule (in practice, a higher-order scheme is applied). If the simulation time points are $\{t_1, \ldots, t_N\}$, then at time t_n, the continuous convolution (1) can be approximated by the following discrete sum:

$$y(t_n) \approx \tilde{y}(t_n) = \sum_{i=1}^{n-1} h(t_n - t_i)x(t_i)\Delta_i, \tag{2}$$

where $\Delta_i = t_{i+1} - t_i$. Over the course of the simulation, n progresses from 1 to N. If we evaluate each sum directly, the total computation time is $O(N^2)$. Unfortunately, we cannot evaluate the convolutions using techniques that require the entire sequence $\{x(t_j)\}$, such as the FFT, since $\tilde{y}(t_n)$ must be evaluated before we know $x(t_j)$ for $j \geq n$. This is because the circuit equations create an implicit dependence of $x(t_j)$ on $\tilde{y}(t_j)$.

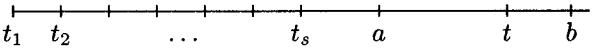

Figure 1: Evaluating equation (5)

3 Review of Approximation Theory

First, we recall a few facts from real analysis [1, 4]. Given a function $f: \mathcal{R} \to \mathcal{R}$ and points $\{c_1, c_2, \ldots, c_p\}$, the unique polynomial of degree $p-1$ which agrees with f at each of these points is

$$P(x) = \sum_{m=1}^{p} u_m(x) f(c_m), \tag{3}$$

where $u_m(x)$ is the mth Lagrange polynomial of degree $p-1$.

$$u_m(x) = \prod_{\substack{k=1 \\ k \neq m}}^{p} \frac{x - c_k}{c_m - c_k}$$

To reduce the interpolation error, when approximating f over an interval $[a, b]$, the $\{c_m\}$ are chosen to be Chebyshev nodes on $[a, b]$.

$$c_m = \frac{a+b}{2} + \frac{b-a}{2} \cos \frac{(2m-1)\pi}{2p}$$

Over this interval, $|f(x) - P(x)|$ is bounded above by

$$\frac{2(b-a)^p}{4^p p!} \max\{\, |f^{(p)}(\xi)|, \text{ where } \xi \in [a, b] \,\}. \tag{4}$$

If the derivatives of f can be bounded and the interval is fixed, then the error decreases exponentially with p.

4 Basic Strategy

To illustrate the basic idea behind our method, we consider the following piece of the sum (2):

$$g(t) = \sum_{i=1}^{s} h(t - t_i)x(t_i)\Delta_i,$$

where $s < n$. For notational convenience, we define $\alpha_i = x(t_i)\Delta_i$.

$$g(t) = \sum_{i=1}^{s} h(t - t_i)\alpha_i \tag{5}$$

We consider the problem of evaluating this sum for t in the interval $[a, b]$, where $[a, b]$ is to the right of $[t_1, t_s]$ (see figure 1). Evaluating the sum directly for any given t requires $O(s)$ operations. Suppose that we apply the approximation in equation (3) to $f(t)$ defined

by $f(t) = h(t - t_i)$. We choose Chebyshev nodes on the interval $[a, b]$ and obtain

$$h(t - t_i) \approx \sum_{m=1}^{p} u_m(t) h(c_m - t_i).$$

Substituting into equation (5) gives

$$g(t) \approx \sum_{i=1}^{s} \left(\sum_{m=1}^{p} u_m(t) h(c_m - t_i) \right) \alpha_i.$$

Interchanging the order of summation yields

$$g(t) \approx \sum_{m=1}^{p} u_m(t) \sum_{i=1}^{s} h(c_m - t_i) \alpha_i.$$

Now note that the inner sum is independent of t; hence we can define

$$\psi_m = \sum_{i=1}^{s} h(c_m - t_i) \alpha_i. \tag{6}$$

The coefficient ψ_m is simply the value $g(c_m)$. The $\{\psi_m\}$ can be evaluated once, in time $O(ps)$, and then stored. Afterwards, evaluating the approximation at any given $t \in [a, b]$ involves only $O(p)$ steps.

$$g(t) \approx \sum_{m=1}^{p} u_m(t) \psi_m$$

(The set of coefficients $\{u_m(t)\}$ can be computed in $O(p)$ operations.) If we need to evaluate $g(t)$ at q points, then computing the sums directly requires $O(qs)$ operations, while the approximation method uses $O(ps + pq)$ steps. *When p is small compared to q and s, the second approach is much faster.*

To quickly evaluate the full sum in equation (2), we use the following strategy. When the simulation has progressed to time t_n, we will evaluate $\tilde{y}(t_n)$ by splitting the sum into two parts. The first part of the sum will cover those time points "close" to t_n, and will be evaluated directly. The second part is for the time points that are "well-separated" from t_n. This part will be evaluated using the approximation method discussed above (see figure 2). Thus, if t_n is within the region $[a, b]$, and if $[a, b]$ contains interpolation nodes $\{c_m\}$, then we define coefficients $\{\psi_m\}$ as in equation (6) and obtain

$$\begin{aligned}
\tilde{y}(t_n) &= \sum_{i=1}^{n-1} h(t_n - t_i) \alpha_i \\
&= \sum_{i=1}^{s} h(t_n - t_i) \alpha_i + \sum_{i=s+1}^{n-1} h(t_n - t_i) \alpha_i \\
&\approx \sum_{m=1}^{p} u_m(t_n) \psi_m + \sum_{i=s+1}^{n-1} h(t_n - t_i) \alpha_i. \qquad (7)
\end{aligned}$$

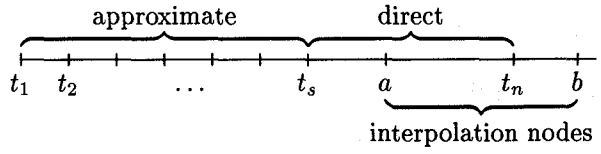

Figure 2: Evaluation of $\tilde{y}(t_n)$

For a given t_n, these last summations can be evaluated in time $O(p + (n - s))$ once the $\{\psi_m\}$ are computed. The computation of the coefficients $\{\psi_m\}$ can be amortized over all of the time points in the interval $[a, b]$.

5 Description of the Algorithm

As the simulation progresses, we must increase s in sum (7) to maintain efficiency. That is, we need to increase the part of the computation that is done approximately. For this purpose, we divide the simulation period into intervals. If the current simulation time lies within interval l, then we will evaluate the part of the sum corresponding to intervals l and $l-1$ directly and the part that represents intervals 1 through $l-2$ approximately. (We evaluate the sum over interval $l-1$ directly since when t_n is near the start of interval l, interval $l-1$ is not well-separated from t_n. This would lead to a poor quality approximation unless the number of interpolation nodes is unreasonably large.) When the simulation advances to interval $l+1$, we update the coefficients $\{\psi_m\}$ to include the part of sum corresponding to interval $l-1$.

More precisely, we write $\{\psi_m^l\}$ to denote the coefficients used for evaluating the sum over intervals 1 through l. The number of interpolation nodes for these coefficients will actually vary from interval to interval, but to avoid notational clutter, we will always just write p. The nodes for interval l will be denoted by $\{c_m^l\}$. The notation $\sum_{i \in l}$ will indicate a sum over time points corresponding to interval l. The coefficient ψ_j^{l+1} is computed by

$$\begin{aligned}
\psi_j^{l+1} &= \sum_{i \in 1 \ldots l+1} h(c_j^{l+1} - t_i) \alpha_i \\
&= \sum_{i \in 1 \ldots l} h(c_j^{l+1} - t_i) \alpha_i + \sum_{i \in l+1} h(c_j^{l+1} - t_i) \alpha_i \\
&\approx \sum_{m=1}^{p} u_m(c_j^{l+1}) \psi_m^l + \sum_{i \in l+1} h(c_j^{l+1} - t_i) \alpha_i
\end{aligned}$$

The coefficients $\{\psi_m^1\}$ are computed directly.

$$\psi_j^1 = \sum_{i \in 1} h(c_j^1 - t_i) \alpha_i$$

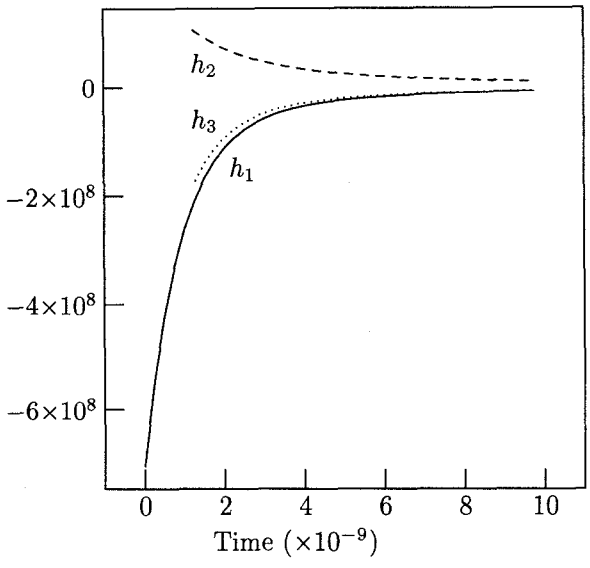

Figure 3: Transmission line impulse responses

For simplicity, we assume that there are exactly q time points in each interval; then the time required to compute the $\{\psi_m^{l+1}\}$ is $O(p^2 + pq)$.

Over the entire simulation, the time spent computing convolutions using the above scheme is accounted for by:

1. N/q sets of coefficients $\{\psi_m^l\}$, each computed in time $O(p^2 + pq)$; plus
2. N convolution sums, each involving $O(2q)$ direct terms and an approximate term that is computed in time $O(p)$.

Hence the total time is $O(N(p^2 + pq)/q + N(2q + p))$.

Now we discuss the selection of interpolation nodes. Note that when we compute $\{\psi_m^l\}$, we must have interpolation nodes on all intervals to the right of interval $l + 1$. This is because information about the sum

$$\sum_{i \in 1 \ldots l} h(t - t_i)\alpha_i$$

is required for all time points in intervals $l + 1$ and onwards. If it were necessary to have p' interpolation nodes *per interval* to attain a given accuracy, then $p = Np'/q$ and the complexity of the method is high. Fortunately, any physically reasonable impulse response $h(t)$ has the property that it gets smoother as t increases. Typical examples are the impulse responses associated with a lossy transmission line; these are shown in figure 3. Based on inequality (4), we know that *fewer interpolation nodes are needed to maintain accuracy in regions where the response is smoother.* In fact, for functions such as those shown, the number of interpolation nodes required grows only logarithmically with N. Thus, we take $p = O(p' \log(N/q))$ and distribute the nodes as shown in figure 4. Also, in the actual implementation, when interpolating at some time t, we use only the appropriate p' nodes surrounding t (so interpolation is really an $O(p')$ operation). We also choose the interval size so that q is about two to three times p'. Then the total time required for convolutions becomes simply $O(Np' \log(N/p'))$. Empirically, we find that p' around 7 is sufficient for single-precision accuracy, and p' around 14 gives double-precision.

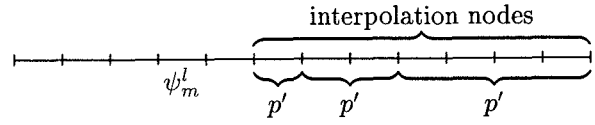

Figure 4: Interpolation node distribution

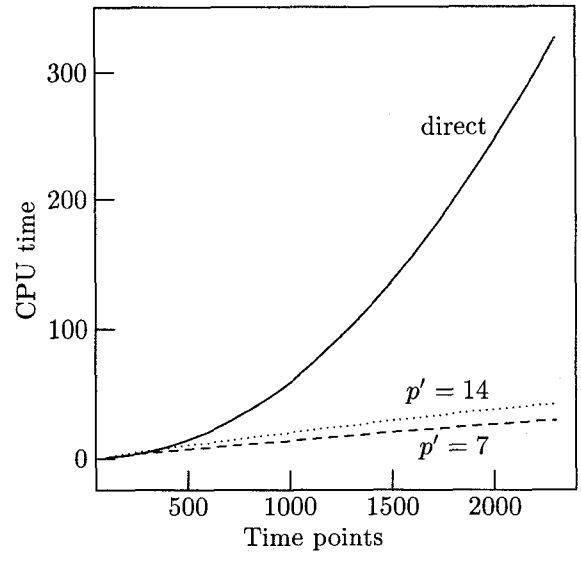

Figure 5: CPU time versus simulation time points

6 Examples

We implemented the above method in a time-domain lossy transmission line model and tested it on a number of examples. Figure 5 shows CPU time versus number of simulation time points for the **mosaic** example from Roychowdhury *et al.* [10] The top curve corresponds to direct convolution, the next is our method with $p' = 14$ (double-precision), and the bottom is our method with $p' = 7$ (single-precision).

As a more realistic example, we considered a low-swing CMOS line driver circuit designed at Bell Labs. The driver contains about one hundred MOSFETs,

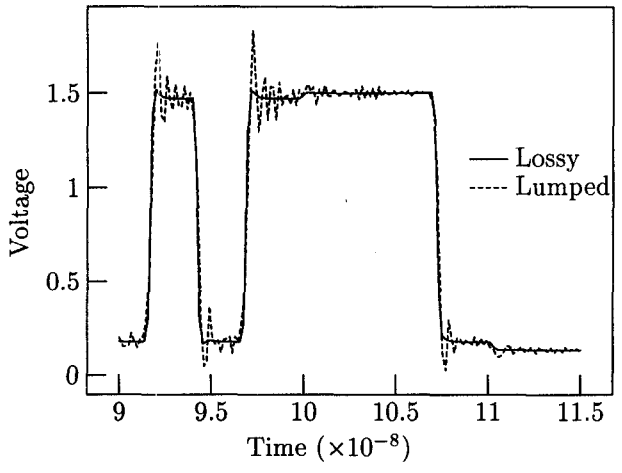

Figure 6: Line driver output waveforms

and the output of the driver is connected through a lossy transmission line to the load. The original circuit design was done with a lumped-element approximation to the transmission line. We ran three simulations, using:

1. the original lumped-element approximation;
2. a lossy transmission line with the convolution computed directly; and
3. a lossy transmission line with the convolution computed by our method with $p' = 14$ (double-precision).

Output waveforms are shown in figure 6. The lumped-element simulation exhibits the artificial ringing that is typical when using such models. Both simulations with a lossy transmission line model agree to double-precision accuracy, but the simulation using our approach is forty percent faster. This example shows only modest speedup since most of the simulation time is spent evaluating the comprehensive MOSFET model. (With our method, about two percent of the total simulation time is spent evaluating the transmission line model.)

7 Conclusions

We have presented an $O(N \log N)$ algorithm for simulating distributed elements such as transmission lines. Our method is based on computing convolutions quickly using approximation theory, and maintains double-precision accuracy. We have implemented a lossy transmission line model using these ideas and demonstrated substantial time savings compared to the direct approach. The scheme is general enough to be applied to a variety of situations, including simulation of devices characterized by measured data. In contrast to other approaches for performing fast convolutions, we do not assume any particular form for the impulse responses: the only requirement is that they become smoother with increasing time.

Acknowledgements

Thad Gabara provided us with the line driver circuit.

References

[1] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions.* National Bureau of Standards, 1964.

[2] B. Alpert and V. Rokhlin. A fast algorithm for the evaluation of Legendre expansions. *SIAM Journal of Scientific and Statistical Computing*, 12(1):158–179, January 1991.

[3] J. E. Bracken, V. Raghavan, and R. A. Rohrer. Interconnect simulation with Asymptotic Waveform Evaluation. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 39(11):869–878, November 1992.

[4] G. Dahlquist and A. Bjorck. *Numerical Methods.* Prentice-Hall, 1974.

[5] H.W. Dommel. Digital computer solution of electromagnetic transients in single and multiphase networks. *IEEE Transactions on Power Apparatus and Systems*, PAS-88(4):388, April 1969.

[6] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data, II. *Applied and Computational Harmonic Analysis*, 2(1):85–100, January 1995.

[7] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, December 1987.

[8] R. Griffith, E. Chiprout, Q. Zhang, and M. Nakhla. A CAD framework for simulation and optimization of high-speed VLSI interconnections. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 39(11):893–906, November 1992.

[9] S. Lin and E. S. Kuh. Transient simulation of lossy interconnects based on the recursive convolution formula. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 39(11):879–892, November 1992.

[10] J. S. Roychowdhury, A. R. Newton, and D. O. Pederson. Algorithms for the transient simulation of lossy interconnect. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, 13(1):96–104, January 1994.

[11] A. Semlyen and A. Dabuleanu. Fast and accurate switching transient calculations on transmission lines with ground return using recursive convolution. *IEEE Transactions on Power Apparatus and Systems*, PAS-94:561–571, 1975.