A Modified Envelope-Following Approach to Clocked Analog Circuit Simulation

Luís Miguel Silveira Jacob White Steven Leeb

Research Laboratory of Electronics
Dept. of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract

In this paper, a modified envelope-following method for simulation of clocked analog circuits is described. The modification makes the envelope-following algorithm more efficient, as unnecessary numerical integration is avoided when computing the envelope of "quasi-algebraic" components in the solution vector. An automatic method for determining the quasi-algebraic solution components is described, and experimental results are given which demonstrate that this modified method reduces the number of computed clock cycles needed to accurately determine the envelope.

1 Introduction

When used to simulate the transient behavior of clocked analog circuits like switching power converters and phase-locked loops, circuit simulation programs like SPICE [1] often employ hundreds of thousands of integration timesteps. This is because the circuit simulation timesteps are constrained to be much smaller than a clock period, but the time interval of interest to a designer can be thousands of clock periods. The high computational cost of simulating such circuits with programs like SPICE has led designers to explore a variety of simulation alternatives, including specialized analog computers [2], and fast simulation techniques based on approximating the circuit using ideal switches and linear elements [3, 4].

Another approach to reducing the computational cost of simulating clocked analog circuits is to exploit the fact that a designer typically is not interested in the details of the node voltage behavior in every clock cycle, but rather is interested in the "envelope" of that behavior. Specifically, we define the "envelope" to be a continuous function derived by interpolating

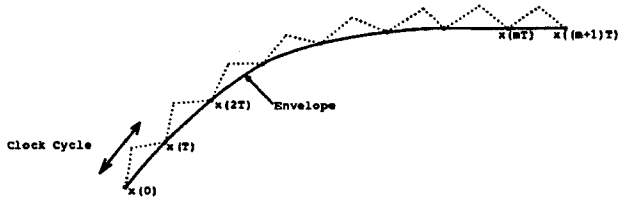

Figure 1: Envelope Definition

the sequence formed by sampling the state every clock period T (See Fig. 1). Note our use of "envelope" is **not** standard. Here, the envelope is not unique given $x(t)$; the envelope generated by interpolating the sequence $x(0+\tau), x(T+\tau), x(2T+\tau), \ldots$ depends on τ.

The key advantage to restricting consideration to the envelope is that if the sequence formed by sampling the state at the beginning of each clock cycle, $x(0), x(T), x(2T), \ldots, x(mT), \ldots$, changes slowly as a function of m, the clock cycle number, then the envelope can be computed accurately by detailed simulation of only every l^{th} clock cycle, where l, referred to as the cycle-step, is large. Computational procedures based on this idea are referred to as **envelope-following** [5] algorithms, and are particularly efficient when used to simulate simplified switching power converters [6, 7]. The method is not effective, however, if there there are states in the system which change rapidly and dramatically due to small changes in much more slowly changing states.

In this paper we describe a modified envelope-following algorithm which circumvents the above problem, and present computational results. In the next section, the basic envelope-following algorithm is described, followed in Section 3 by a description of our modified version. In Section 4, the standard and modified envelope-following algorithms are compared with

CH3026-2/91/0000/0020/$01.00 © 1991 IEEE

standard methods on several circuit examples. Finally, in Section 5, conclusions and acknowledgments are given.

2 Envelope-Following

Most clocked analog circuits can be described by a system of differential equations of the form

$$\frac{d}{dt}\boldsymbol{p}(\boldsymbol{x}(t),\boldsymbol{u}(t)) + \boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t)) = 0, \qquad (1)$$

where $\boldsymbol{x}(t) \in \mathbf{R}^N$, the state, is the vector of capacitor voltages and inductor currents, $\boldsymbol{u}(t) \in \mathbf{R}^M$ is the vector of input sources, $\boldsymbol{p}(\boldsymbol{x}(t),\boldsymbol{u}(t)) \in \mathbf{R}^N$ is the vector of capacitor charges and inductor fluxes, and $\boldsymbol{f}(\boldsymbol{x}(t),\boldsymbol{u}(t)) \in \mathbf{R}^N$ is the vector of resistive currents and inductor voltages.

If the state $\boldsymbol{x}$ is known at some time t_0, it is possible to solve Eqn. (1) and compute the state at some later time t_1. In general, one can write

$$\boldsymbol{x}(t_1) = \boldsymbol{\phi}(\boldsymbol{x}(t_0), t_0, t_1) \qquad (2)$$

where $\boldsymbol{\phi} : \mathbf{R}^n \times \mathbf{R} \times \mathbf{R} \rightarrow \mathbf{R}^n$ is a state transition function for the differential equation.

The straight-forward approach to computing the envelope of the solution to Eqn. (1) is to numerically compute $\boldsymbol{x}(t)$ for all t and then to sample this computed solution at $\boldsymbol{x}(0)$, $\boldsymbol{x}(T)$, $\boldsymbol{x}(2T)$,... to construct the envelope. If the envelope is *smooth enough*, then it will be possible to approximately represent an interval of sample points, $\boldsymbol{x}((m-1)T), \boldsymbol{x}(mT) \ldots \boldsymbol{x}((m+l)T)$ with a low order polynomial in the cycle number. For example, if over $l+1$ cycles the envelope is well approximated by a straight line, then

$$\boldsymbol{x}((m+l)T) - \boldsymbol{x}(mT) \approx l[\boldsymbol{x}(mT) - \boldsymbol{x}((m-1)T)]. \quad (3)$$

The term $[\boldsymbol{x}(mT) - \boldsymbol{x}((m-1)T)]$ can be thought of, imprecisely, as the derivative of the envelope at $\boldsymbol{x}(mT)$, in which case Eqn. (3) is loosely analogous to solving a differential equation by forward-Euler. Following that analogy, l is then the *cycle-step* for the integration method.

As circuit problems are stiff, the cycle-step will be severely restricted due to stability considerations unless an implicit integration method is used to compute the envelope. The simplest implicit envelope-following scheme is based on backward-Euler and is given by

$$\boldsymbol{x}((m+l)T) - \boldsymbol{x}(mT) = l[\boldsymbol{x}((m+l)T) - \boldsymbol{x}((m+l-1)T)] \qquad (4)$$

where $\boldsymbol{x}(mT)$ is known from the previous cycle-step, $\boldsymbol{x}((m+l-1)T)$ must be computed, and $\boldsymbol{x}((m+l)T)$ is determined from $\boldsymbol{x}((m+l-1)T)$ by solving Eqn. (1) over one cycle. Using the state transition function defined in Eqn. (2), the relation between $\boldsymbol{x}((m+l)T)$ and $\boldsymbol{x}((m+l-1)T)$ can be written as

$$\boldsymbol{x}((m+l)T) = \boldsymbol{\phi}(\boldsymbol{x}((m+l-1)T), (m+l-1)T, (m+l)T). \qquad (5)$$

Using this relation in Eqn. (4) yields the nonlinear algebraic equation

$$\boldsymbol{\phi}() - \boldsymbol{x}(mT) = l[\boldsymbol{\phi}() - \boldsymbol{x}((m+l-1)T)] \qquad (6)$$

from which $\boldsymbol{x}((m+l-1)T)$ can be determined given $\boldsymbol{x}(mT)$. Note that the arguments for $\boldsymbol{\phi}$ are omitted in (6) for brevity, but are the same as in (5).

An iterative Newton's method can be applied to solving (6) for $\boldsymbol{x}((m+l-1)T)$, in which case the iteration update equation is

$$\boldsymbol{J_F}(\boldsymbol{x}^k)[\boldsymbol{x}^{k+1} - \boldsymbol{x}^k] = -\boldsymbol{F}(\boldsymbol{x}^k), \qquad (7)$$

where k is the Newton iteration count, $\boldsymbol{F}(\boldsymbol{x}^k)$ is given by

$$\boldsymbol{F}(\boldsymbol{x}^k) = \boldsymbol{\phi}() - \frac{l}{l-1}\boldsymbol{x}^k + \frac{1}{l-1}\boldsymbol{x}(mT) \qquad (8)$$

and $\boldsymbol{J_F}$ is given by

$$\boldsymbol{J_F}(x) = \frac{\partial}{\partial \boldsymbol{x}}\boldsymbol{\phi}() - \frac{l}{l-1}\boldsymbol{I}_N \qquad (9)$$

where $\boldsymbol{I}_N$ is the identity matrix of size N.

The most time-consuming computation in this Newton iteration is evaluating $\boldsymbol{J_F}$ and $\boldsymbol{F}$, which involves computing the state transition function and its derivative. The state transition function can be evaluated by numerically integrating Eqn. (1) from $(m+l-1)T$ to $(m+l)T$ given $\boldsymbol{x}((m+l-1)T)$. The derivative of the state transition function, referred to as the sensitivity matrix, represents the sensitivity of $\boldsymbol{x}((m+l)T)$ to perturbations in $\boldsymbol{x}((m+l-1)T)$ and can be computed with a small amount of additional work during the numerical integration, as is described in [8]

3 Removing Quasi-Algebraic Variables

If the backward-Euler envelope-following algorithm in Section 2 is used unmodified, the cycle-step will be constrained by the component of $\boldsymbol{x}$ with the fastest changing envelope. This can be unnecessarily conservative, as components of $\boldsymbol{x}$ which have rapidly changing envelopes are likely to be nearly algebraic functions of other, more slowly changing components, at least over the time scale of one clock period. That is,

these nearly, or quasi-, algebraic components of $\boldsymbol{x}$ can be computed directly, and therefore envelopes associated with quasi-algebraic nodes need not be computed with a formula like Eqn. (4).

Which components of $\boldsymbol{x}$ are quasi-algebraic can be determined using the same sensitivity matrix already required to solve Eqn. (6) with Newton's method. To see this, note that a component x_i in $\boldsymbol{x}$ is quasi-algebraic in one clock period if all components of $\boldsymbol{x}$ are insensitive to x_i's value at the beginning of a period. By definition, entry (i,j) in the sensitivity matrix represents the sensitivity of $x_i((m+l)T)$ to perturbations in $x_j((m+l-1)T)$. Therefore, x_i is a quasi-algebraic component if the i^{th} column of the sensitivity matrix is nearly zero.

Now let the components of $\boldsymbol{x}$ be divided into two vectors: $\boldsymbol{x}_s$, the vector of true states, and $\boldsymbol{x}_a$, the quasi-algebraic vector. Then the sensitivity matrix can be reordered so that

$$\frac{\partial \phi(\boldsymbol{x}_s(T), \boldsymbol{x}_a(T))}{\partial(\boldsymbol{x}_s(0), \boldsymbol{x}_a(0))} = \begin{bmatrix} \frac{\partial \boldsymbol{x}_s(T)}{\partial \boldsymbol{x}_s(0)} & \frac{\partial \boldsymbol{x}_s(T)}{\partial \boldsymbol{x}_a(0)} \\ \frac{\partial \boldsymbol{x}_a(T)}{\partial \boldsymbol{x}_s(0)} & \frac{\partial \boldsymbol{x}_a(T)}{\partial \boldsymbol{x}_a(0)} \end{bmatrix} \quad (10)$$

By the definition of a quasi-algebraic component, the second column of $\frac{\partial \phi(\boldsymbol{x}_s(T), \boldsymbol{x}_a(T))}{\partial(\boldsymbol{x}_s(0), \boldsymbol{x}_a(0))}$ is nearly zero. The standard envelope-following algorithm can then be applied to a subset of the circuit variables, using as the sensitivity matrix the $\frac{\partial \boldsymbol{x}_s(T)}{\partial \boldsymbol{x}_s(0)}$ block diagonal submatrix. As the sensitivity matrix is updated every cycle, that $\frac{\partial \boldsymbol{x}_s(T)}{\partial \boldsymbol{x}_a(0)}$ and $\frac{\partial \boldsymbol{x}_a(T)}{\partial \boldsymbol{x}_a(0)}$ remain small can be verified, and a decision can be made about which variables should be considered quasi-algebraic for subsequent cycle computations. This provides an automatic algorithm for determining quasi-algebraic components.

In Algorithm 1, we give the complete modified envelope-following algorithm. Note that in Algorithm 1, at the beginning of every cycle-step, the quasi-algebraic components, $\boldsymbol{x}_a$, are computed from the state components, $\boldsymbol{x}_s$, and this involves computing a *DC* solution with the state components held fixed.

ALGORITHM 1.

Nitswit Modified Envelope-Follower

```
Divide x into x_s (States) and
  x_a (quasi-algebraic) using the
  Sensitivity matrix.
m = 0
While mT < STOPTIME {
   Select the cycle-step l
   Predict a first guess, x_s^0((m+l-1)T)
   Compute x_a^0((m+l-1)T) from x_s^0((m+l-1)T).
   Numerically integrate Eqn. (1) from
     (m+l-1)T to (m+l)T to compute
     x^0((m+l)T) and ∂x^0((m+l)T)/∂x((m+l-1)T)
   Compute J_F(x^0((m+l-1)T)) as in Eqn. (9)
   Redivide x into x_s (States) and
     x_a (quasi-algebraic) using the new
     Sensitivity matrix.
   Set k=0
   Until Newton Converges {
      Solve the Newton update equation for
        x_s^{k+1}((m+l-1)T).
      Compute x_a^{k+1}((m+l-1)T) from
        x_s^{k+1}((m+l-1)T).
      Numerically integrate
        Eqn. (1) from (m+l-1)T to (m+l)T
        to compute x^k((m+l)T)
      }
   m = m+l
}
```

4 Implementation and Results

Both the standard and a modified version of the envelope-following method have been implemented in the *Nitswit* simulation program. The program uses a trapezoidal-rule based envelope-following algorithm with local-truncation error cycle-step control. In Table 1, we compare the cpu time required to simulate the start-up transient from four different circuits by classical direct methods, the standard envelope-following algorithm and our modified algorithm. The circuits presented are: a resonant converter *quasi* [9], an open-loop buck converter circuits, *dbuck*, a closed loop converter, *closed* and a switched capacitor filter, *scop*. In each case, the clocking is provided by a user-defined source. As can be seen from the table, the envelope-following method can be very efficient, particularly when the simulation interval is long compared to the clock period. In particular, from the results presented it is clear that the standard envelope following algorithm is very efficient when simulating open-loop circuits.

The results obtained when comparing envelope-following to classical methods for a closed-loop buck converter *closed* does not produce equally encouraging results. The difficulty simulating the closed-loop converter is that it includes control circuitry which rapidly responds to small changes in the converter output. However, variables associated with the controller are

Circuit	N	Cycles	Clas.	Std EF	Mod EF
quasi	7	200	188	69.4 (33)	16.5 (13)
scop	13	200	156	65 (30)	27.6 (15)
dbuck	4	1000	359	34.5 (50)	29.0 (48)
closed	5	600	79	47 (124)	10.8 (31)

Table 1: CPU Time (in seconds on a SUN4 260) Comparisons for Classical and Standard Envelope-Following Simulation versus Modified Envelope Following. The number of cycles shown corresponds to the simulation interval divided by the clock period. For the envelope following approaches, the number of effectively simulated cycles is also shown.

quasi-algebraic, and therefore the modified algorithm performs substantially better. Note also that the results in Table 1 shows that modified envelope following is always faster than the standard envelope-following, due to the reduction in the number of computed cycles. Most noticeably, for the most difficult example, namely the closed loop converter, a speedup of a factor of over four is obtained over standard envelope following, and this makes the modified envelope following algorithm almost an order of magnitude faster than the classical direct approach.

5 Conclusions and Acknowledgments

In this paper it is shown that a modified envelope-following approach to the simulation of switching power and filter circuits can provide substantial speed improvements over classical simulation methods even for closed-loop converters. Several aspects of the modified method are still under investigation; of particular importance is finding faster techniques for updating the algebraic variables when leaping over some cycles. In addition, the effectiveness of the envelope following is somewhat dependent on where the cycle boundaries are placed, and an automatic selection method is desirable.

The authors would like to thank Professor John Kassakian, Professor James Kirtley, Professor Martin Schlecht, Professor George Verghese at M.I.T.'s Laboratory for Electromagnetic and Electronic Systems for their assistance in understanding the switching power simulation problem, and Prof. A. Sangiovanni-Vincentelli and Prof. A. R. Newton at U. C. Berkeley, for their suggestions. In addition, the authors are indebted to Ken Kundert from Cadence Design Systems for his suggestions and for his contributions to the Nitswit program. This work was supported by the Defense Advanced Research Projects Agency contract N00014-87-K-825, the National Science Foundation contract MIP-8858764, grants from IBM and Digital Equipment Corporation, and the Portuguese INVOTAN committee.

References

[1] L. W. Nagel, "SPICE2: A computer program to simulate semiconductor circuits," Tech. Rep. ERL M520, Electronics Research Laboratory Report, University of California, Berkeley, Berkeley, California, May 1975.

[2] J. Kassakian, "Simulating power electronic systems – a new approach," *Proceedings of the IEEE*, vol. 67, October 1979.

[3] C. J. Hsiao, R. B. Ridley, H. Naitoh, and F. C. Lee, "Circuit-oriented discrete-time modeling and simulation for switching converters," *IEEE Power Electronics Specialists' Conf. Rec.*, 1987.

[4] S. C. Fang, *et al*, "Switcap: A switched-capacitor network analysis program – part 1: Basic features," *IEEE Circuits Syst. Mag.*, vol. 5, September 1983.

[5] L. Petzold, "An efficient numerical method for highly oscillatory ordinary differential equations," *SIAM J. Numer. Anal.*, vol. 18, June 1981.

[6] K. Kundert, J. White, and A. Sangiovanni-Vincentelli, "An envelope-following method for the efficient transient simulation of switching power and filter circuits," in *International Conference on Computer Aided-Design*, (Santa Clara, California), pp. 446–449, October 1988.

[7] J. White and S. Leeb, "An envelope-following approach to switching power converter simulation," *IEEE Trans. on Power Electronics*, vol. 6, April 1991.

[8] T. Aprille and T. Trick, "Steady-state analysis of nonlinear circuits with periodic inputs," *Proceedings of the IEEE*, January 1972.

[9] L. F. Casey and M. F. Schlecht, "A high frequency, low volume, point-of-load power supply for distributed power systems," *IEEE Power Electronics Specialists' Conf. Rec.*, 1987.