Error Control and Limit Cycle Elimination in Event-Driven Piecewise Linear Analog Functional Models

Byong Chan Lim, *Member, IEEE*, and Mark Horowitz, *Fellow, IEEE*

***Abstract*—Real number modeling of analog circuits in hardware description languages (HDLs) has become more common as a part of mixed-signal SoC validation. We propose two methods that both improve the fidelity and simulation speed, and make the event-driven, piecewise linear (PWL) analog functional models easier to write. First we use the accuracy set by users to dynamically determine when a new output segment should be emitted, which is computed without any iteration. This capability allows designers to trade accuracy for simulation speed of analog models without any time-consuming model calibration/error estimation, and creates models which generate events only when needed to maintain output accuracy. We next extend this method to eliminate limit-cycle oscillations that occur when simulating circuits with continuous-time feedback in a discrete-time event simulator. Handling this feedback efficiently allows the user to create the system model from simpler component models. The performance of this modeling approach is demonstrated on various analog filter models, an operational amplifier, and a high-speed, wireline transceiver system, and is 3.1× faster than an optimally-chosen, fixed-time step simulation for the transceiver.**

***Index Terms*—Feedback, limit cycle, linear system, mixed-signal system, modeling, piecewise linear, Verilog.**

I. Introduction

REAL number modeling of analog circuits in HDLs such as (System)Verilog and VHDL is widely deployed for the system verification today [1]–[13]. As analog and digital systems are becoming more tightly coupled, design teams need to verify the functionality of the overall mixed-signal system at the system level, which requires fast, high-fidelity analog functional models. In this paper, we leverage a PWL waveform approximation to represent analog signals and propose a way to dynamically schedule the events for approximating the signal waveform to PWL segments with a well controlled error bound. This dynamic scheduling of events eliminates the need to find the largest time step that gives acceptable accuracy, facilitating model creation and improving simulation performance.

Analog functional models commonly use oversampled, discrete-time representations of the analog signals. They sample the analog signals at a rate much higher than Nyquist, and represent the dynamic behavior of a system by using a discrete time model (e.g., IIR, FIR filters) [2], [13]. Although oversampling seems to be inefficient in modeling RF systems, the carrier is generally not sampled; instead the baseband signal is oversampled and the carrier frequency information is appended to the signal [3], [10]. While this method has been widely used, it is not free from issues. Since this signal representation is quantized in time, it has difficulty in modeling the effects of timing jitter, and it forces one to either run the entire analog chain at a single clock rate, or create specialized models to connect analog sections that run at different rates.

To address many of these issues, Jang *et al.* presented a new way of representing an analog signal as a sum of pre-defined basis functions and carrying the parameters of the functions over a wire [11], [12]. They map an analog circuit to a linear or weakly non-linear dynamical system and perform analog filtering operation in Laplace *s*-domain. This work relies on the fact that most analog circuits can be modeled by a nearly linear model in some transformed domain (voltage, phase, frequency, etc.) [14]–[18]. Since a signal waveform is completely expressed as an analytic equation, the output signal needs to be updated only when the system's inputs change, which greatly reduces the number of events that the system needs to handle. While this method is extremely promising, it currently has two limitations that cause us to use a different approach. First, we need to handle piecewise linear models which require internal events to handle the change of linear regions, and second we need to model continuous-time analog feedback systems. This feedback cannot be handled in their current system, since every input change causes an output change, which then causes another input change.

Another approach to address the issues of analog functional models is to use a PWL waveform which was proposed by Cottrell in 1990 [1] and has been recently rediscovered [7]–[9]. PWL signals have both value and slope information so that values between samples can be interpolated, allowing the model to compute the effect of sampling clock jitter even if the signal has a coarse time step. However, the PWL model still requires regular evaluation of the PWL signal to control the approximation error, and the time step for the evaluation is typically limited by the steepest curvature of the signal. Liao *et al.* point out that the required time step can be inferred from the bandwidth of the component being modeled [8]. However, the relationship between this time step and the waveform error

Manuscript received July 24, 2015; revised October 28, 2015; accepted November 20, 2015. This paper was recommended by Associate Editor B.-D. (Brian) Liu.

The authors are with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: bclim@stanford.edu; horowitz@stanford.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2015.2512699

depends on the system being modeled, so the time step must be set by the model writer. This task is more difficult when modeling adjustable analog circuits (e.g., a circuit where its bandwidth changes with control voltage input). In addition, the regular evaluation of PWL segments is redundant when the curvature of the signal waveform is not large, slowing down simulation.

Our analog model builds upon a piecewise linear system model to easily handle both nonlinearity and continuous-time feedback loops, which leads us to represent an analog signal as a PWL waveform. Instead of evaluating a waveform at a fixed rate, the analog model is completely event-driven and controls time steps dynamically at runtime for a given error tolerance. This time-step control allows a model writer to uniformly trade off the simulation accuracy for speed without requiring any effort (model calibration, i.e., approximation error vs. time step). This capability is especially useful in system-level validation environment, since the model parameters of analog blocks keep changing in the design phase and the required accuracy changes as well. Initial regression tests are looking for gross errors, while later ones look for more subtle effects. This time-step control allows the system to get accurate enough results, while controlling for the effect on the overall simulation speed.

While the idea of adaptive time step control is very old, forming the basis of SPICE simulators [19], and even being proposed for use in functional models in the mid 1990s by Pichon *et al.* [7], our use and implementation are novel. First, we use it primarily to make model writing cleaner, and to give the user an obvious knob for adjusting the computation needed to model the analog components. Second, since we know we are modeling piecewise linear systems as we show in Section II, we can compute the time-step directly, with neither iteration nor time integration.

Having made this change, we notice that limit cycle behavior of analog functional models with a continuous-time analog feedback loop continues to generate redundant events. The frequency of the limit cycle oscillation is much higher than that of the event generation in our model, and thus the oscillation should be suppressed. Our model dynamically finds these limit cycles and removes them in simulation.

Section II briefly reviews the PWL modeling approach used to describe the dynamic behavior of an analog circuit. Section III explains the proposed dynamic time step control. Limit cycle behavior in PWL approximation and its solution is explained in Section IV. Section V demonstrates the performance of the proposed method and how one can easily describe complex behaviors with piecewise linear system models with examples.

II. Background

In PWL modeling, a signal is represented in a piecewise linear format in which value and slope are updated at discrete intervals in an event-driven simulator (e.g., SystemVerilog/ VHDL simulators [20]). In SystemVerilog, for example, a piecewise linear signal can be defined using *struct* as shown in Listing 1, and its member variables are typically updated at uniform intervals set by designers. Sometimes, the event time, $t0$, is omitted (e.g., the work in [8]).

```
typedef struct {
  real y;     // signal offset
  real slope; // signal slope
  real t0;    // time offset
} pwl;  // pwl datatype
```

Listing 1. Definition of a piecewise linear (*pwl*) waveform using *struct* in SystemVerilog.

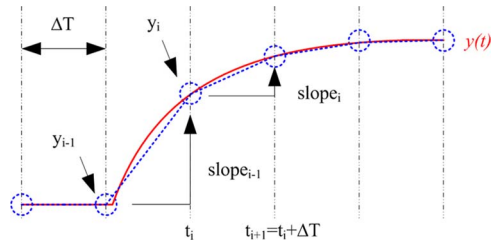

Fig. 1. Piecewise linear approximation (blue dotted line) of a continuous-time waveform $y(t)$ with fixed time step evaluation.

```
parameter etol = 0.001; // user-defined error tolerance
time dt=10; // user-defined fixed time step
real out0, out1;
pwl out; // output PWL signal
reg wakeup=1'b0;

// schedule an event for every 'dt'
initial forever wakeup = #(dt) ~wakeup;

always @(new_input or wakeup) begin
  t0 = `get_time; // in second
  if (new_input)
    update parameters; // (a, b, ci, pi) of y(t);
  yi = out.yi + out.slope*dt; // current offset value
  yi_1 = y(t0+dt); // next offset value
  slope_new = (yi_1-yi)/dt; // possible new slope
  // yi_p: projected output w/ previous PWL segment
  yi_p = yi + out.slope*dt;
  if (abs(yi_1-yi_p) > etol) // filtering output event
    out = '{yi, slope_new, t0}
end
```

Listing 2. Pseudocode of PWL waveform approximation with a fixed time step dt. The $y(t)$ is a system's response being approximated.

To represent the dynamic behavior of an analog circuit, the circuit is modeled as a piecewise linear dynamical system [8], [9], [11], [12]. Since the mathematical form of each PWL segment is a ramp function, the output response to a ramp input is calculated, which is a sum of complex exponentials and a ramp function

$$y(t) = a + b \cdot t + \sum_i c_i \cdot e^{-p_i \cdot t}. \quad (1)$$

Since zero and pole frequencies could be complex numbers, the actual response in time-domain should take the real part of (1).

In writing a PWL functional model, the number of parameters in (1) is already known while their values (a, b, c_i, and p_i) vary in simulation. Typically, events are scheduled with a fixed time step (Δt) to update the output waveform as the approximation is depicted in Fig. 1 and its SystemVerilog pseudocode is shown in Listing 2. The statements enclosed by *always* are executed either when a new input comes in or for

```
parameter etol = 0.001; // user-defined error tolerance
time dt; // dynamically controlled time step
pwl out; // output PWL signal
event wakeup;

always @(new_input or wakeup) begin
  t = 'get_time; // in second
  yi = out.yi + out.slope*(t-out.t0) // evaluate offset
  if (new_input)
    update parameters; // (a, b, ci, pi) of y(t);
  calculate dt; // dt = f(y(t), y''(t), etol)
  slope_new = (y(t+dt)-yi)/dt;
  err = abs(dt*(out.slope-slope_new));
  if (err>etol) // filtering output event
    out = '{yi, slope_new, t}
  ->> #(dt) wakeup; // schedule an event
end
```

Listing 3. Pseudocode of PWL waveform approximation with the proposed method.

every Δt (dt in Listing 2) fired by the *wakeup* signal. This means that the output PWL segment (*out*) is recomputed every Δt. At time t_0, the difference in the expected values of *out* and $y(t)$ at $t_0 + \Delta t$ is calculated and compared to the user-defined absolute error tolerance, e_{tol}. If the difference is less than e_{tol}, the PWL segment *out* will not be updated. On the other hand, if the error is larger than e_{tol}, *out* is updated to have a new PWL segment. Although some output events are filtered out, the evaluation of the error is always executed for every Δt.

This filtering of small output changes is essential to simulate a continuous-time feedback loop. Without it, any input changes will propagate around the loop indefinitely, so the simulator will hang [8]. Outlawing this type of external feedback by forcing all feedback loops to be internal to a functional model is a bad idea for two reasons. First it breaks the modularity of the functional model: it might require one to create a new model for a combination of blocks that have already been modeled. Second, this type of feedback is important to model explicitly if the underlying amplifier is piecewise linear, since this arrangement will accurately model the closed loop effects of these nonlinearities (e.g., slew limiting and output saturation). Writing a closed loop model that contains these effects can be quite complex.

Notice that this "infinite loop" problem does not happen in oversampled models, or in any system where the feedback wraps a discrete time component such as a phase locked loop and a switched-capacitor circuit. In these systems there is a clock, or at least a time tick delay between an input and the output responding to the input's change.

III. The Proposed PWL Modeling Method

To remove the task of setting the time step by a model writer and to minimize the number of events for given error tolerance, we control the time step Δt dynamically in simulation by evaluating the curvature of the response $y(t)$. As shown in Listing 3, the proposed method is completely event-driven; the *always* block is executed only when either a new input comes in or an event-scheduling signal $wakeup$ is triggered to update a PWL segment for error control after Δt (dt in Listing 3). The rest of this section describes the details of the Δt calculation.

A. Dynamic Time Step Control

When approximating a function $y(t)$ to a piecewise linear segment for the interval $t_0 \le t \le t_0 + \Delta t$, the approximation error err is bounded by

$$|err| \le \frac{1}{8} \cdot \Delta t^2 \cdot \max_{t_0 \le t \le t_0 + \Delta t} |y''(t)| \tag{2}$$

using Rolle's theorem, assuming the function $y(t)$ has a continuous second derivative $y''(t)$ [21]. The needed time step Δt for the error tolerance e_{tol} is then given by

$$\Delta t_{(t=t_0)} = \sqrt{\frac{8 \cdot e_{\text{tol}}}{\max_{t_0 \le t \le t_0 + \Delta t} |y''(t)|}} \tag{3}$$

where e_{tol} is $|err|$ in (2). Since $\max |y''(t)|$ for the interval $t_0 \le t \le t_0 + \Delta t$ is generally not a closed-form equation, calculating Δt requires iterations, resulting in slower simulation speed. To calculate Δt without any iteration, we leverage the fact that we know the form of our system response, which is given in (1). We can use our knowledge of the complex exponential terms $(c_i \cdot e^{-p_i \cdot t})$ to bound the approximation error.[1]

Let us start with a single decaying exponential function:

$$y(t) = \alpha \cdot e^{-\beta \cdot t}, \quad \alpha \in \mathbb{R} \quad \text{and} \quad \beta \in \mathbb{R}^+. \tag{4}$$

Since $y''(t)$, the second derivative of $y(t)$, is a scaled version of $y(t)$, $|y''(t)|$ is also a decaying function and thus the maximum value of $|y''(t)|$ should be at $t = t_0$ for the interval $t_0 \le t \le t_0 + \Delta t$, resulting in a closed form of (3):

$$\Delta t_{(t=t_0)} = \sqrt{\frac{8 \cdot e_{\text{tol}}}{|y''(t_0)|}} = \sqrt{\frac{8 \cdot e_{\text{tol}}}{|\alpha \cdot \beta^2 \cdot e^{-\beta \cdot t_0}|}}. \tag{5}$$

This allows one to calculate the next wakeup time $(t_0 + \Delta t)$ without any iteration. Once Δt is found, the slope of the output segment is updated to $(y(t_0 + \Delta t) - y(t_0))/\Delta t$.

The error control capability of the proposed method is demonstrated on a single-pole filter model where Listing 5 in the Appendix shows its SystemVerilog code. The response to 40 ps ramp input is shown in Fig. 2. The events (marked with filled circles) are also shown and the interpolated residual errors to its analytic expression are within the given error tolerance, e_{tol}. The model generates less events to reduce simulation runtime when e_{tol} is relaxed from 1 mV to 10 mV.

Equation (5) is easily extended when the response is a sum of (complex) exponentials, $y_0(t)$ and $y_1(t)$, as follows:

$$y_0(t) = \sum_i \alpha_i \cdot e^{-\beta_i \cdot t}, \quad \alpha_i \in \mathbb{R} \quad \text{and} \quad \beta_i \in \mathbb{R}^+ \tag{6}$$

$$y_1(t) = \sum_i \alpha_i \cdot \cos(\gamma_i \cdot t + \phi_i) \cdot e^{-\beta_i \cdot t},$$

$$\alpha_i, \gamma_i, \phi_i \in \mathbb{R} \quad \text{and} \quad \beta_i \in \mathbb{R}^+. \tag{7}$$

[1] An offset term and a ramp function in (1) do not create any approximation error in piecewise linear waveform approximation. Thus those two terms are ignored in the analysis.

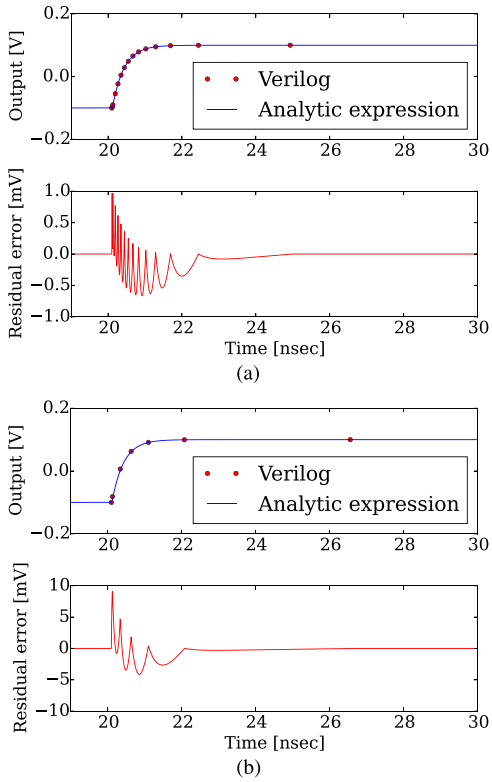

Fig. 2. Comparison of ramp responses of a single-pole filter model (pole at 500 MHz): PWL waveform with the proposed method (events are marked with filled circles) vs. the response from its ideal transfer function in Laplace s-domain. (a) $e_{\rm tol} = 1$ mV and $\max(|\text{residual error}|) = 0.97$ mV, and (b) $e_{\rm tol} = 10$ mV and $\max(|\text{residual error}|) = 9.08$ mV.

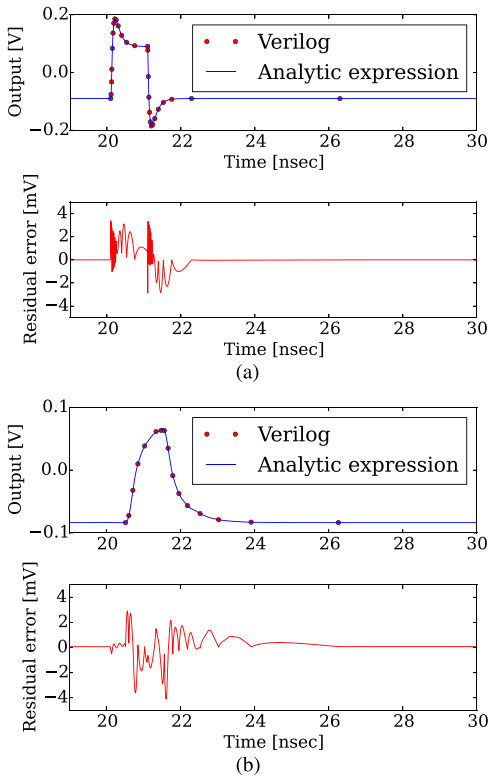

Fig. 3. Comparison of pulse responses of PWL waveform with the proposed method (events are marked with filled circles and $e_{\rm tol} = 5$ mV) vs. the responses from their ideal transfer functions in Laplace s-domain: (a) CTLE filter model (zero at 550 MHz, and poles at 1.2 GHz and 3.4 GHz) responses and $\max(|\text{residual error}|) = 3.38$ mV, and (b) channel filter model responses and $\max(|\text{residual error}|) = 4.11$ mV.

In these cases, we simply calculate a conservative Δt to avoid any iteration. Assuming the response function is $y_0(t)$ as in (6), $|y_0''(t)|$ is then given by

$$|y_0''(t)| = \left| \sum_i \alpha_i \cdot \beta_i^2 \cdot e^{-\beta_i \cdot t} \right|. \tag{8}$$

In this case, a new function $g(t)$ is defined by taking the absolute value of each term in (8) as follows.

$$g(t) = \sum_i \left| \alpha_i \cdot \beta_i^2 \cdot e^{-\beta_i \cdot t} \right| \tag{9}$$

where each term of $g(t)$ is now a decaying exponential function. By triangular inequality ($|z_1 + z_2| \le |z_1| + |z_2|$), $g(t)$ is the upper bound of $|y_0''(t)|$. Therefore, a conservative time step can be obtained by replacing $|y_0''(t)|$ with $g(t)$ in (3) as follows:

$$\Delta t_{(t=t_0)} = \sqrt{\frac{8 \cdot e_{\rm tol}}{g(t_0)}}. \tag{10}$$

The same principle holds for $y_1(t)$ in (7) by equating $|\cos(\gamma_i \cdot t + \phi_i)|$ to one.

Note that, in practice, the response of an analog circuit is mostly dominated by a few poles. Therefore, (9) does not increase simulation time much compared with (8).[2] This approximation of the time step calculation is typically not an issue for a system with complex poles and zeros [i.e., (7)]. Often, a complex system model with many important poles and zeros arises from modeling a system with delay. This system model can be greatly simplified by modeling the delay explicitly in the function model [22]. In RF systems, a baseband-equivalent model is frequently used such that the baseband signal does not contain a carrier signal component [i.e., high-frequency sinusoidal terms in (7)] [3], [10].

Two filter models frequently used in high-speed link application are also examined with $e_{\rm tol} = 5$ mV; one is a continuous-time linear equalizer (CTLE) having a real zero and two real poles, and the other is a physical channel having a real pole, a real zero, three complex poles and their conjugates, and three complex zeros and their conjugates. As the responses are shown in Fig. 3, the residual errors of the simulated waveforms to their analytic models are well bounded yet conservative as expected.

Note that unlike circuit simulators this method does not perform any time integration and thus does not suffer from any numerical stability issue. At each event ($t = t_i$), the current value y_i is evaluated with the current PWL segment, and then the slope of a new PWL segment ($slope_i$) is updated by evaluating the next time step (Δt_i) and $y(t_i + \Delta t_i)$ as follows:

$$y_i = y_{i-1} + slope_{i-1} \cdot (t_i - t_{i-1}) \tag{11}$$

$$slope_i = \frac{y(t_{i+1}) - y_i}{\Delta t_i} \tag{12}$$

[2] In practice, one can take only the largest term in (9) for speed although the approximation error could be slightly larger than $e_{\rm tol}$.

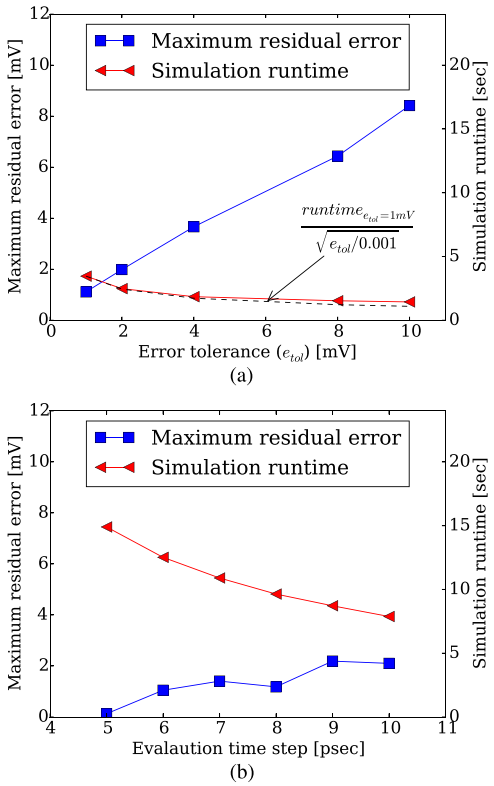

Fig. 4. Maximum residual error and simulation runtime of the channel model used in Fig. 3(b) vs. (a) user-defined error tolerance with the proposed method and (b) evaluation time step with the fixed time step method.

where $t_{i+1} = t_i + \Delta t_i$. Therefore, there is no truncation error at $t = t_{i+1}$ unlike time integration methods, and the error between events is always bounded by the proposed method.

Of course, this method cannot guarantee that all the approximation errors in a system will be less than $e_{\rm tol}$, since it is a local method. A small approximation error of a signal could be amplified by a subsequent circuit. For example, imagine a low-pass filter followed by a high-pass filter where the two filters have the same bandwidth; the overall system is an identity system. An out-of-band input signal to the low pass filter could completely disappear if its output signal is less than an error tolerance. In fact, this problem exists for all functional modeling systems with waveform approximation regardless of time step calculation method. However, this error amplification is not an issue since the overall system being validated needs to be robust to small variations in the components. Therefore, pointing out a high error amplification path would not be a bad result.

The generation of these filter models is easily automated. The event scheduling block is always the same, and the only parts which need to be customized are the routines to get $y(t)$ and its derivatives. These parts can be generated automatically from a Laplace s-domain transfer function specified by a user. The corresponding time-domain expressions to a ramp input are easily obtained by symbolic math and then plugged into a filter model template to create the model in SystemVerilog.

B. Comparison With Fixed-Time Step Control

Fig. 4(a) shows the trade-off between maximum residual error and simulation speed for our variable time step method and Fig. 4(b) shows the same trade-off for the fixed time step method for the channel model shown in Fig. 3(b). The input pattern to the channel is 50k symbols from a 7-bit pseudorandom bit sequence (PRBS) pattern generator running at 2.5 Gbps.

Fig. 4(a) shows that maximum residual error is less than and proportional to $e_{\rm tol}$. As expected, the simulation runtime is proportional to $1/\sqrt{e_{\rm tol}}$, since Δt is proportional to $\sqrt{e_{\rm tol}}$ in (10). As shown in Fig. 4(b), for a fixed-time step the error dependence on time step is more complex, and it takes three times the simulation time compared with the variable time step method to achieve the same quality.

This three times speed up understates the performance advantage of the variable time step control. For example, as mentioned in Section III-A, the complex poles and zeros used to model the channel in Fig. 3(b) simply create a phase shift and thus are modeled as an explicit time delay [22]. Using a simplified channel model with explicit delay, the proposed time step control method is roughly 6× faster than the fixed time step method. Similarly, using the original channel model and slowing the symbol rate to 250 Mbps, the speed of the proposed method is 15× faster than the fixed time step control as the proposed method creates less events compared with the fixed time step control. In addition to the performance advantage, using variable time steps simplifies model creation by avoiding the need for model calibration. A considerable amount of time is required to choose a proper time step of each module in writing models with the fixed time step method, and the time steps need to be recalibrated when the system's characteristics (i.e., parameters in the models) are changed. As the size of a system being modeled gets larger, error calibration can become more troublesome and time-consuming; these procedures disappear with the proposed variable time step control.

IV. Limit Cycle in Feedback System

To prevent infinite loop in continuous-time feedback systems, a gatekeeper similar to the work in [8] is implemented, but is capable of handling the proposed dynamic event control. The gatekeeper can be placed explicitly on a feedback path or integrated into one of the modules in the loop. Whenever a new input comes in, it will compare the current values of its input and output. If the difference between them is greater than the given error tolerance, it will propagate the input to the output. Otherwise, the input change will be ignored and a new event is scheduled by projecting the time when the error is expected to be greater than the error tolerance.

While the infinite loop is easily prevented, there still remains a limit cycle oscillation in steady-state which is inherent in an event-driven model. Even though the circuit is stable and converges, the use of an error tolerance creates a dead zone which will cause oscillations when wrapped in a feedback loop. For example, Fig. 5 shows a first-order system model of a feedback amplifier, and the limit cycle oscillation of the model is observed at the output ($V_{\rm OUT}$) as shown in Fig. 6. The characteristics of a limit cycle oscillation are different from those of the oscillation manifested in an unstable system; the amplitude of the limit cycle oscillation is less than $e_{\rm tol}$ and its frequency is the closed-loop bandwidth of a feedback system.

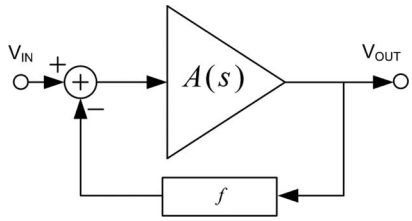

Fig. 5. Feedback amplifier model with i) a feed-forward amplifier where its transfer function $A(s)$ is $A_o/(1+s/w_{p1})$ ($A_o = 200$ and $w_{p1} = 2\pi \cdot 2.5$ Mrad/s) and ii) a feedback network where $f = 0.5$.

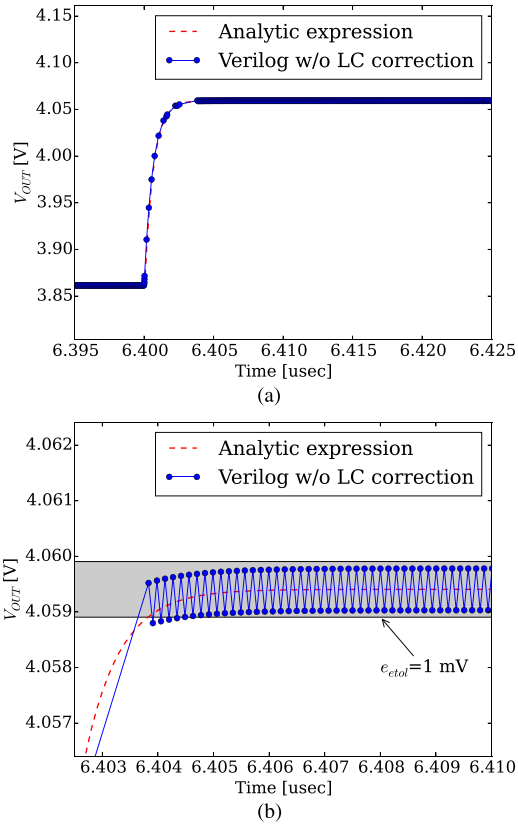

Fig. 6. Limit cycle oscillations observed at the output ($V_{\rm OUT}$) of the feedback amplifier model shown in Fig. 5 in steady-state condition [$e_{\rm tol} = 1$ mV, and events are marked with filled circles)] without the proposed limit cycle corrector: (a) a ramp response and (b) the settling behavior of the ramp response in (a).

This limit cycle behavior does not cause any functional problem, since the signal is still bounded by the error tolerance. However, it slows down the simulation speed, since it creates redundant events in steady-state condition (i.e., events occurred after about 6.404 μs in Fig. 6). If the steady-state condition lasts for T seconds and the period of the limit cycle is p, $2 \cdot T/p$ superfluous events will be created. The oscillation is also observed in both an oversampled model and a PWL model with fixed time step method. In those models, however, the oscillation does not degrade simulation speed much since the oscillation period is typically longer than the time step.

To fix this limit cycle problem, we provide the gatekeeper with a limit cycle corrector, thereby forcing the output to have a settled value with zero slope and eventually having all the nodes

```
// in: input signal of the gatekeeper
// out: output signal of the gatekeeper

// To: DC loop gain
// specified by users
parameter real To = 250;
parameter real etol_Rk=1-To/(1+To);


always @(in or wakeup) begin
  // Return ratio
  Rk = in.slope/out.slope;

  // check range (sdiff=smax-smin)
  if (new_input) begin
    if (in.slope>0) begin
      smax = in.val0;
      w1 = abs(in.slope);
    end
    else if (in.slope<0) begin
      smin = in.val0;
      w2 = abs(in.slope);
    end
    sdiff = abs(smax - smin);
  end

  // detecting limit cycle condition
  if ( abs(1+Rk)<etol_Rk ) begin
    // settle the output
    out = '{(smax+smin)/2,0,t0};
    lc_detect <= 1'b1;
  end
  else
    perform normal gatekeeper operation

always @(any_external_inputs) begin
  lc_detect <= 1'b0;
end
```

Listing 4. Pseudocode for resolving limit cycle issue in a continuous-time feedback system.

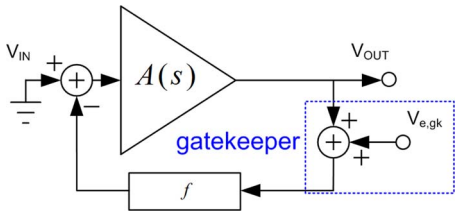

Fig. 7. A system model in limit cycle oscillations: a transfer function of $V_{\rm OUT}$ to the error $V_{e,gk}$ caused by a gatekeeper.

in the loop be settled. Listing 4 shows the partial pseudocode of the gatekeeper to resolve limit cycle issue.

Fig. 7 shows a continuous-time system model in limit cycle oscillations where the error caused by the error tolerance of a gatekeeper is modeled as an input $V_{e,gk}$ to the system. The transfer function of $V_{\rm OUT}$ to $V_{e,gk}$ is then given by

$$\frac{V_{\rm OUT}}{V_{e,gk}} = -\frac{T(s)}{1+T(s)} \tag{13}$$

where $T(s)$ is the frequency-dependent loop gain (i.e., $A(s) \cdot f$) of the feedback system. In event-driven simulations, on the other hand, limit cycle is detected by monitoring the return ratio (R_k) of the feedback loop which is defined as follows:

$$R_k = in.slope/out.slope \tag{14}$$

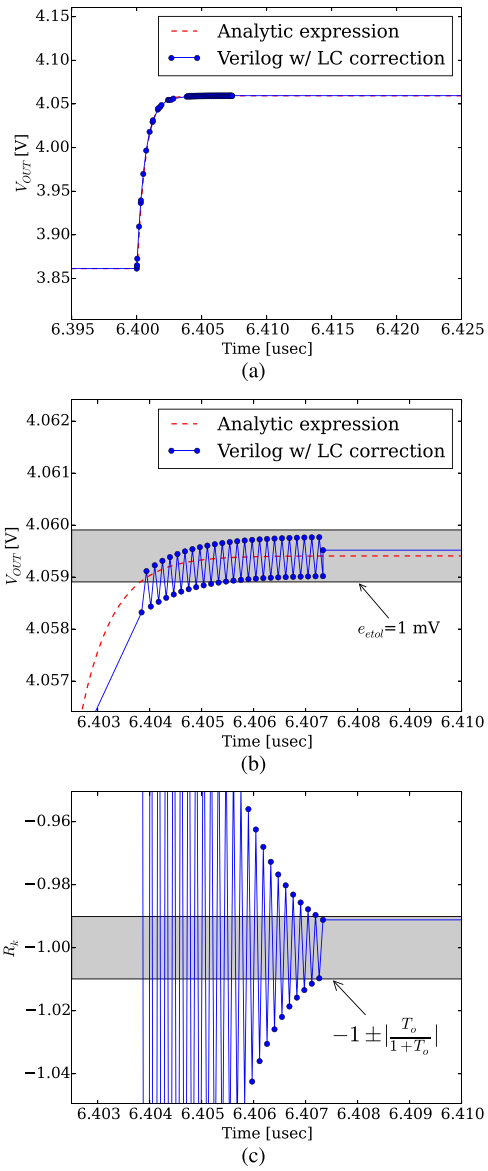

Fig. 8. Comparison of ramp responses of the feedback system in Fig. 5 from the Verilog simulation with the proposed limit cycle correction and the system's analytic model: (a) ramp responses, (b) the settling behaviors of the ramp responses in (a), and (c) return ratio (R_k) from Verilog simulation.

where *in.slope* and *out.slope* are the slopes of input and output PWL segments of a gatekeeper, respectively. Ideally, R_k should be -1 in limit-cycle oscillations. However, in practice it is slightly off from -1 because of the finite loop gain $T(s)$ in (13). Therefore, the condition on detecting limit cycle oscillations is given by

$$|1 + R_k| \leq 1 - \left[\frac{T(s)}{1 + T(s)}\right] \tag{15}$$

from (13) and (14). Since a continuous-time signal being approximated should be in steady state (i.e., $s = 0$), the condition is simplified to

$$|1 + R_k| \leq 1 - \left[\frac{T_o}{1 + T_o}\right] \tag{16}$$

where T_o is $T(s)$ at $s = 0$ (e.g., $T_o = 100$ in Fig. 5).

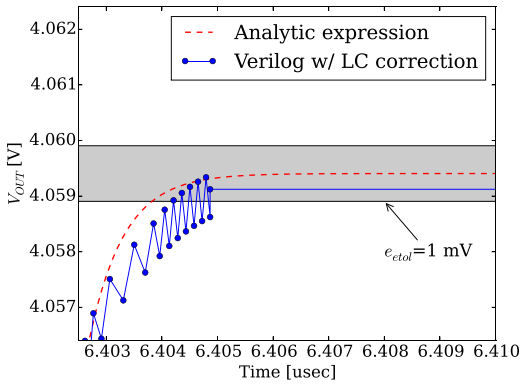

Fig. 9. Settling behavior of the feedback system in Fig. 5 using the proposed limit cycle correction where $1 - |T_o/(1 + T_o)|$ is set to 0.33 instead of 0.0099.

Once a limit cycle is detected in simulation, an expected output value in steady-state condition is calculated from the two peak values in limit cycle oscillations. The maximum value (smax) is updated when the input signal ascends, i.e., $in.slope > 0$, while the minimum value (smin) is updated when the input signal descends, i.e., $in.slope < 0$. If the system is in limit cycle condition, the slope $(out.slope)$ of the output PWL signal is set to zero and its offset value $(out.y)$ is set to the average value of the maximum (smax) and the minimum (smin) values as follows:

$$out.y = \frac{\mathrm{smax} + \mathrm{smin}}{2}. \tag{17}$$

With the proposed limit cycle correction model, we are able to completely stop the limit cycle oscillations as shown in Fig. 8. In the example, the simulation speed without limit-cycle oscillation is more than 1000× faster than the speed with the oscillation, if the steady-state condition lasts for 1 ms. The gatekeeper stops the oscillation when R_k is met the condition in (16). As observed in Fig. 8(b) and (c), the limit cycle corrector stops the oscillation of V_{OUT} when the exact response from its analytic expression settles within the specified error tolerance value (i.e., e_{tol}) and this oscillation is detected by observing R_k.

Note that the error of the computed value in (17) to a steady-state value of an analytic model could be larger than e_{tol}, if a larger tolerance value than one in (16) is chosen. Since this error is amplified by a gain stage, i.e., $A(s)$, and the bandwidth of a feedback loop is finite, this error can make the signals around the loop to wander over time, causing another limit-cycle oscillations. However, the frequency of the new oscillations will typically be a few hundreds times lower than the closed-loop bandwidth of the feedback system. Therefore, this signal wandering should not degrade simulation speed much.

Of course, it is possible to avoid signal wandering completely. In Listing 4, when the oscillation is initially detected, the lc_detect signal is asserted and can be broadcasted to another blocks in the loop so as to force all the signals on the loop to be frozen. For normal operation, this lock condition is released and the system will be activated when any external input to the feedback system is triggered. For example, Fig. 9 shows the settling behavior of the feedback system in Fig. 5

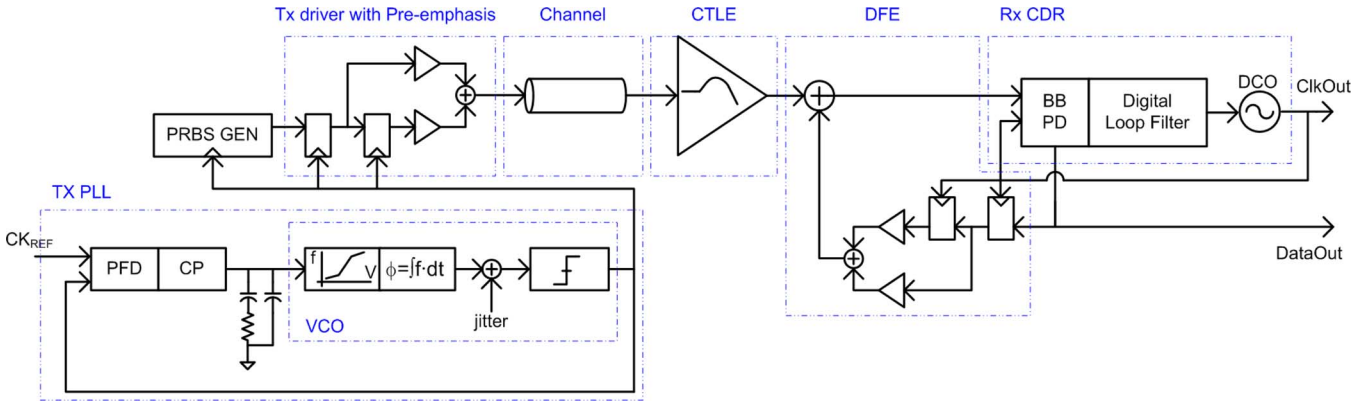

Fig. 10. Wireline transceiver system example.

by setting $1 - |T_o/(1+T_o)|$ to 0.33 instead of 0.0099. The limit-cycle oscillations disappear quickly compared with the simulated waveform in Fig. 8. However, this method has two drawbacks. One is the approximation error could be larger than $e_{\rm tol}$. The other is that the model could not be pin accurate to its circuit implementation, since broadcasting lc_detect needs port punching of the models.

V. Experimental Results

The performance of the proposed modeling method is demonstrated with two examples. All the models are written in SystemVerilog, and the model simulations run on a Linux machine with Intel i5-4570 CPU (3.2 GHz).

A. High-Speed Serial Link

The first example is a complete wireline transceiver system shown in Fig. 10. In the transmitter, a random data stream is generated by a 7-bit PRBS pattern generator which is synchronized to the 5 GHz output clock from a charge-pump phase locked loop (PLL). In the PLL, random jitter (1 ps rms) is easily added to the output of the voltage controlled oscillator (VCO) in phase domain. The generated stream is then transmitted to the receiver through a two-tap pre-emphasis FIR filter and the physical channel used in Fig. 3(b). In the receiver, the front-end circuit is the CTLE used in Fig. 3(a), followed by a decision feedback equalizer (DFE) with a two-tap FIR filter. The clock data recovery circuit in the receiver is a bang-bang phase detector with a digital PLL.

Each block of the system is modeled with the proposed technique and Fig. 11(a) shows the eye diagram measured at the output of the DFE. The $e_{\rm tol}$ values of the phase-domain VCO model and the loop filter in the PLL are set to 0.02 radian and 1 mV, respectively. The $e_{\rm tol}$ value of both the channel and CTLE is set to 1 mV. The simulation runtime is 13.5 s to exercise 50 k symbols at 5 Gbps, and the runtime is reduced to 7.0 s when the $e_{\rm tol}$ value of both the channel and CTLE is relaxed from 1 mV to 10 mV.

For comparison, the PWL models using a fixed time step were also built for the same transceiver system. To have $e_{\rm tol}$ of 1 mV, the time step values of the channel and CTLE are set to 6 ps and 1.3 ps, respectively, after brute-force search. With this fixed time step method, the simulation runtime is 42.4 s, which is still 3.1× longer than the proposed method, while its eye diagram shown in Fig. 11(b) is similar to the diagram shown in Fig. 11(a).

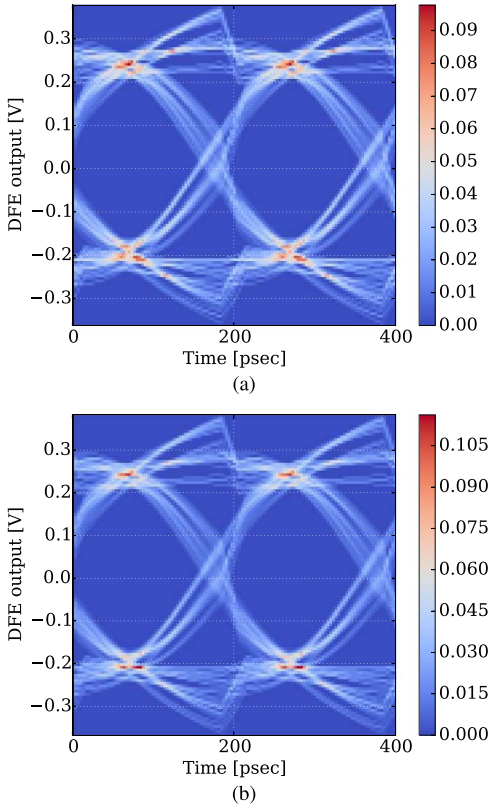

Fig. 11. Eye diagram (50k symbols at 5 Gbps) strobed at the DFE output of the transceiver shown in Fig. 10: (a) with the proposed method and (b) with the fixed time step method.

B. Pseudo-Class-AB OTA

The ability to handle both external analog feedback and nonlinear behavior is demonstrated with a pseudo-class-AB operational transconductance amplifier (OTA) shown in Fig. 12,

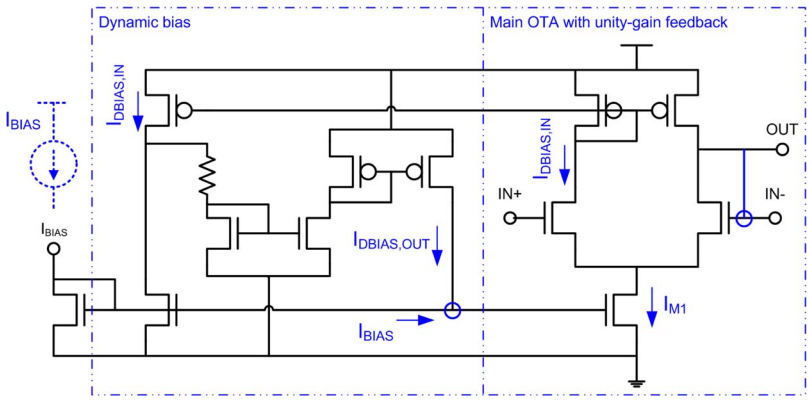

Fig. 12. Pseudo-class-AB operational transconductance amplifier.

similar to the circuit in [23].[3] The dynamic bias current generation circuit boosts the bias current (I_{M1}) to the OTA circuit when the input voltage difference $V_{\mathrm{IN}+} - V_{\mathrm{IN}-}$ to OTA is positive, while the bias current to the OTA remains almost the same as the current supplied through the input I_{BIAS} when $V_{\mathrm{IN}+} - V_{\mathrm{IN}-}$ is negative. Because of the dynamic bias current control as well as the inherent nonlinearity of the OTA, the circuit shows a strongly nonlinear behavior in transient.

Mapping a circuit to a linear dynamic system guides us in writing an accurate functional model. The circuit is partitioned into two smaller blocks (i.e., the main OTA with a bias current control input and a dynamic bias current circuit) and each block is modeled as a piecewise linear system. The main OTA circuit is a first-order linear system where its parameters (transconductance, pole, slew rate) vary with operating points of the bias current and the common-mode voltage of differential inputs. The bias control circuit is another piecewise linear system which adjusts the bias current of the first system. Since the complete circuit forms two external feedback loops (i.e., a loop around the bias control circuit and a unity-gain feedback loop), two gatekeepers are placed on the feedback paths to prevent limit-cycle oscillations. The parameters of the system to operating points are characterized at transistor level using a SPICE-like simulator and their mapping functions are built as look-up tables in SystemVerilog HDL.

Fig. 13 compares the circuit simulator and SystemVerilog simulator waveforms of the OTA. Both rise and fall transition waveforms of the output voltage V_{OUT} match between circuit and model simulation results. Limit-cycle oscillations are observed for both transitions when the circuit settles, but they are eliminated by the proposed gatekeeper and the Verilog model does not create any more event. At steady state, gain mismatch is slightly larger when $V_{\mathrm{IN}+}$ is 2.5 V compared with $V_{\mathrm{IN}+}$ is 1.4 V, since secondary effects (e.g., the dependency of the output resistance on the input common-mode voltage in the main OTA) are not modeled.

In this example, the model simulation is only 14 times faster than the circuit simulation. Event-driven simulation is

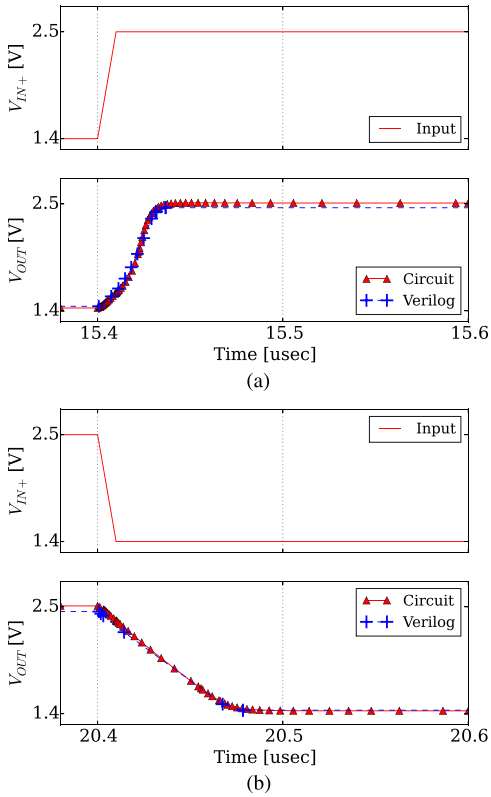

Fig. 13. Comparison of transient waveforms ($V_{\mathrm{IN}+}$ and V_{OUT}) between circuit simulation and Verilog simulation of the pseudo-class-AB OTA in Fig. 12 ($t_{r,V_{\mathrm{IN}+}} = t_{f,V_{\mathrm{IN}+}} = 10$ nsec, VDD $= 3.3$ V): (a) rise transient waveforms, and (b) fall transient waveforms.

less efficient for external feedback loop modeling. Closed-loop models typically show a few hundreds to a few thousands times speedup over circuit simulation. However, the bandwidth of this kind of analog feedback loop is generally much lower than those of other parts in a complete system. Therefore, although the speed improvement of this small block is only tens of times faster, the overall speed improvement is typically much higher.

VI. Conclusion

Analog functional modeling using PWL waveforms is attractive for mixed signal validation because of its ability to

[3]Generally, to drive a large load with small static current, this kind of circuit is designed to boost the bias current in both rise and fall transient as described in [23]. However, the circuit in this example is modified for demonstration purpose to only adjust the slew rate of the rising edge.

```
module sp_filter #(
  parameter etol = 0.001, // error tolerance
  parameter w1 = 2*3.141592*500e6 // pole in radian
) ( input pwl in, output pwl out);

timeunit 1fs;
timeprecision 1fs;

// get time unit and assign it to TU
real TU; initial $get_timeunit(TU);

// wires
event  wakeup;  // event signal
real t0;  // time offset
real t_cur;   // current time
real dTr;  // dT of PWL waveform in real value format
time dT=1; // dT in time format
time dTm;
time t_prev;
real out_cur; // current output signal value
real out_nxt;  // out at (t_cur+dT) for pwl output data
real out_slope; // out slope
real out0;  // initial value of out when a new input
    comes in
real a0, b0, c0; // parameters of y(t)
real err;

// a new input comes in
always @(in.y or in.slope) begin
  t0 = $realtime*TU;
  out_cur = eval_pwl(out, t0);
  out0 = out_cur;
  a0 = in.y - in.slope/w1;
  b0 = in.slope;
  c0 = -in.y + in.slope/w1 + out0;
  t_prev = $realtime;
  dT = 0;
  ->> wakeup;
end

// schedule an event to update PWL segments
always @(wakeup) begin
  dTm = $realtime - t_prev;
  if (dT==dTm) begin
    t_prev = $realtime;
    t_cur = $realtime*TU;
    out_cur = eval_pwl(out, t_cur);
    dTr = calculate_Tintv_spf(etol, t_cur-t0);
    dTr = max(TU,min(dTr,1));
    out_nxt = yt(t_cur-t0+dTr);
    out_slope = (out_nxt-out_cur)/dTr;
    dT = time'(dTr/TU);
    err = abs(dTr*(out.slope-out_slope));
    if (err>=etol)
      out = '{out_cur, out_slope, t_cur};
    ->> #(dT) wakeup;
  end
end

// calculating y(t)
function real yt;
input real t;
begin
  return a0 + b0*t + c0*exp(-w1*t);
end
endfunction

// Caluating dT
function real calculate_Tintv_spf;
input real etol, t;
real abs_f2max;
begin
  abs_f2max = abs(c0)*w1**2*exp(-w1*t);
  return sqrt(8.0*etol/abs_f2max);
end
endfunction

// Evaluate a PWL segment
function real eval_pwl (pwl in, real t);
  return in.y + in.slope*(t-in.t0);
endfunction

endmodule
```

Listing 5. A single-pole filter model in SystemVerilog.

model feedback and non-linear behavior using piecewise linear models. By taking advantage of the ability to compute the closed form output for each input change, we can dynamically control the time step while bounding the maximum error of the approximated output waveform. While this system works well in models without external analog feedback, analog feedback causes limit cycle oscillations that reintroduce fixed time events, slowing down simulation. To avoid this issue, we add a limit cycle corrector model to our framework, which detects and eliminates small oscillations within the error tolerance. The resultant PWL modeling framework can handle most mixed signal systems, and is easier to use since users no longer need to estimate the time step to put in each model. The performance of this framework is more than three times faster than existing PWL approaches.

APPENDIX
A SINGLE-POLE FILTER MODEL IN SYSTEMVERILOG

A linear system having a pole w_1 can be expressed in Laplace s-domain as follows:

$$T(s) = \frac{Y(s)}{X(s)} = \frac{1}{1 + s/w_1}. \tag{18}$$

A ramp input $X(s)$ in Laplace s-domain is $a/s + b/s^2$, and the output response $Y(s)$ is a product of $T(s)$ and $X(s)$ in Laplace s-domain. The corresponding time-domain equation $y(t)$ after taking inverse Laplace transform of $Y(s)$ is a sum of forced and natural responses, which is given by

$$y(t) = a - \frac{b}{w_1} + b \cdot t + \left\{ -a + \frac{b}{w_1} + y(0^-) \right\} \cdot e^{-w_1 \cdot t} \tag{19}$$

where $y(0^-)$ is an initial condition of $y(t)$.

Listing 5 is a complete SystemVerilog code of a single-pole filter model with the proposed dynamic time step control method. A pole frequency location is parameterized as $w1$.

REFERENCES

[1] R. Cottrell, "Event-driven behavioural simulation of analogue transfer functions," in *Proc. Eur. Design Autom. Conf. (EDAC)*, Mar. 1990, pp. 240–243.

[2] W. Hartong and S. Cranston, "Real valued modeling for mixed signal simulation," Cadence Design Systems, 2009.

[3] J. David, "Radio receiver mixer model for event-driven simulators to support functional verification of RF-SoC wireless links," in *Proc. IEEE Int. Beh. Model. Simul. Conf. (BMAS)*, Sep. 2010, pp. 42–47.

[4] Y. Wang, C. Van-Meersbergen, H.-W. Groh, and S. Heinen, "Event driven analog modeling for the verification of PLL frequency synthesizers," in *Proc. IEEE Beh. Model. Simul. Workshop (BMAS 2009)*, Sep. 2009, pp. 25–30.

[5] S. Joeres, H.-W. Groh, and S. Heinen, "Event driven analog modeling of RF frontends," in *Proc. IEEE Beh. Model. Simul. Workshop (BMAS 2007)*, Sep. 2007, pp. 46–51.

[6] R. Staszewski, C. Fernando, and P. Balsara, "Event-driven simulation and modeling of phase noise of an RF oscillator," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 4, pp. 723–733, Apr. 2005.

[7] F. Pichon, S. Blanc, and B. Candaele, "Mixed-signal modeling in VHDL for system-on-chip applications," in *Proc. Eur. Design Test Conf. (ED TC 1995)*, Mar. 1995, pp. 218–222.

[8] S. Liao and M. Horowitz, "A Verilog piecewise-linear analog behavior model for mixed-signal validation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 8, pp. 2229–2235, Aug. 2014.

[9] M.-J. Park, H. Kim, M. Lee, and J. Kim, "Fast and accurate event-driven simulation of mixed-signal systems with data supplementation," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Sep. 2011, pp. 1–4.
[10] J.Chen, "A modeling methodology for verifying functionality of a wireless chip," in *Proc. IEEE Beh. Model. Simul. Workshop (BMAS 2009)*, Sep. 2009, pp. 96–101.
[11] J.-E. Jang, M.-J. Park, D. Lee, and J. Kim, "True event-driven simulation of analog/mixed-signal behaviors in SystemVerilog: A Decision-Feedback Equalizing (DFE) receiver example," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Sep. 2012, pp. 1–4.
[12] J. E. Jang, M. Park, and J. Kim, "An event-driven simulation methodology for integrated switching power supplies in SystemVerilog," in *Proc. 50th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, May 2013, pp. 1–7.
[13] F. A. Nothaft, L. Fernandez, S. Cefali, N. Shah, J. Rael, and L. Darnell, "Pragma-based floating-to-fixed point conversion for the emulation of analog behavioral models," in *Proc. 2014 IEEE/ACM Int. Conf. Comput.-Aided Design*, 2014, pp. 633–640.
[14] J. Kim *et al.*, "Leveraging designer's intent: A path toward simpler analog CAD tools," in *Proc. IEEE CICC*, Sep. 2009, pp. 613–620.
[15] J. Kim *et al.*, "Variable domain transformation for linear PAC analysis of mixed-signal systems," in *Proc. IEEE/ACM ICCAD*, Nov. 2007, pp. 887–894.
[16] J. Kim *et al.*, "Stochastic steady-state and AC analyses of mixed-signal systems," in *Proc. ACM/IEEE DAC*, Jul. 2009, pp. 376–381.
[17] M. Horowitz *et al.*, "Fortifying analog models with equivalence checking and coverage analysis," in *Proc. ACM/IEEE DAC*, 2010, pp. 425–430.
[18] B. C. Lim *et al.*, "An efficient test vector generation for checking analog/mixed-signal functional models," in *Proc. ACM/IEEE DAC*, Jun. 2010, pp. 767–772.
[19] K. S. Kundert, *The Designer's Guide to SPICE and Spectre*. Norwell, MA, USA: Kluwer Academic, 1995.
[20] Accellera, "SystemVerilog LRM 3.1a," 2004.
[21] J.Stewart, *Calculus*. Boston, MA, USA: Cengage Learning, 2007.
[22] A. Morched, B. Gustavsen, and M. Tartibi, "A universal model for accurate calculation of electromagnetic transients on overhead lines and underground cables," *IEEE Trans. Power Del.*, vol. 14, no. 3, pp. 1032–1038, 1999.
[23] M. G. Degrauwe, J. Rijmenants, E. A. Vittoz, and H. J. De Man, "Adaptive biasing CMOS amplifiers," *IEEE J. Solid-State Circuits*, vol. 17, no. 3, pp. 522–528, 1982.

Byong Chan Lim (M'08) received his Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 2012, and is currently a Research Associate in the same department. His research interests include bioelectronics, mixed-signal circuit design and methodologies for improving the productivity of mixed-signal SoC design and validation. Before pursuing the Ph.D. degree, he had worked on developing various analog IPs and mixed-signal SoCs such as high-speed link, clocking circuits, image processor, and so on in industry.

Mark Horowitz (F'00) is the Yahoo! Founders Professor at Stanford University, Stanford, CA, USA, and was chair of the Electrical Engineering Department from 2008 to 2012. He co-founded Rambus, Inc. in 1990 and is a Fellow of the ACM and a member of the National Academy of Engineering and the American Academy of Arts and Science. Dr. Horowitz's research interests are quite broad and span using EE and CS analysis methods to problems in molecular biology to creating new design methodologies for analog and digital VLSI circuits.