Fast and Accurate Event-Driven Simulation of Mixed-Signal Systems with Data Supplementation

Myeong-Jae Park, Hanseok Kim, Minbok Lee, Jaeha Kim
Inter-university Semiconductor Research Center, Seoul National University, Seoul, Korea

Abstract- **This paper presents a methodology of simulating behaviors of complex mixed-signal systems such as phase-locked loops (PLLs) and high-speed I/O interfaces on an event-driven HDL simulator like SystemVerilog. Continuous-time signal events as well as voltage and timing noises are accurately modeled without relying on fine time steps, based on a technique called data supplementation. That is, rather than representing a signal just as a series of time-value pairs, additional information is annotated using the *struct* construct in SystemVerilog. Prototype models for a 2-GHz PLL and a 2-Gbps high-speed series I/O system including a 2.9dB-loss channel at 1GHz demonstrate 50× and 80× faster simulation speeds, respectively, with the same or better accuracy compared with the conventional time-step based models.**

I. Introduction

To verify functionality and evaluate performance of complex mixed-signal systems such as high-speed serial I/O interfaces, one needs an efficient behavioral simulator that is fast in handling a large number of digital logic events and also accurate in simulating analog signals. While numerous solutions exist, including those based on Verilog-AMS [1] or Matlab/Simulink [2][3], they tend to sacrifice one or the other. For instance, event-driven simulation approaches [2][3] are poor at expressing continuous-time signals while fixed time-step based approaches [4] are inefficient for large-scale systems. This paper presents a technique called data supplementation that can accurately model various analog behaviors without any speed loss on an event-driven logic simulator such as SystemVerilog.

Efforts to model analog behaviors accurately tend to slow down simulation speeds because they can result in a large number of finely-spaced events. For example, when modeling a data receiver that samples a noisy incoming signal triggered by a jittery clock, a straightforward way is to use a very fine time step to express the variations in voltage and timing.

However, accuracy need not cost speed penalty. For instance, a time-step-based PLL simulator in [5] relaxed this need for fine time steps by implicitly expressing the actual cross times of a discrete-level clock using continuous levels. In this paper, we extend this approach while making it more explicit. That is, we supplement additional information to the signal event so that it can help preserve the accuracy of the signal being modeled. For instance, we can tag the actual crossing time to the clock transition event to express it explicitly without relying on the time step precision. And we demonstrate that this approach can be further extended to modeling other analog behaviors such as the output signal of a PLL loop filter and that of a linear filter or lossy transmission line.

The rest of this paper is organized as follows. Section II describes the details of applying this data supplementation technique in a SystemVerilog simulator. Section III and IV illustrates their use in modeling the behaviors of a PLL and a linear filter. Section V discusses the experimental results on the speed and accuracy of the proposed technique compared with the conventional approaches.

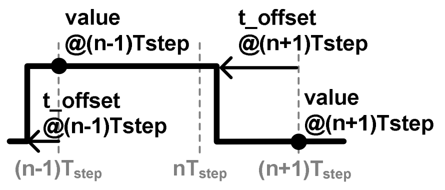
II. Accurate Modeling of Clock and Analog Signals with Data Supplementation

SystemVerilog was chosen as our baseline event-driven simulator for two reasons. First, digital functional models verified with this simulator can be synthesized directly without further modification. This is an advantage over non-HDL behavioral simulators such as Matlab/Simulink [2][3] or C++[5]. Second, SystemVerilog supports the use of *struct* type variables as the module input/output ports, allowing us to attach auxiliary information with various data types without altering the module port interfaces.

The first example of data supplementation is the expression of accurate clock waveform, for which a *struct* data type named xbit is defined, as illustrated in Fig. 1. A supplemental variable t_offset in the structure indicates the actual time instant of the clock's transition with the time offset from the event's time stamp. For instance, if a clock signal's value switches from 0 to 1 at time 10, a t_offset value of 0.4 indicates that the actual transition occurred at time 9.6. Therefore, the resolution in expressing the clock transition timing is virtually infinite, without requiring fine time steps that slow down simulation speeds.

While this approach can express finer timing resolution than the simulator time step, it should be noted that the xbit events that belong to the same time step but have different t_offset values may not necessarily occur in a chronological order. An example is illustrated in Fig. 2. A sampler is capturing the input data value at the instant of the input clock's rising edge.

```
typedef struct {
   bit value;
   real t_offset;
} xbit;
```



(a) (b)

Fig. 1. Supplementing transition time information to express accurate clock waveforms: (a) SystemVerilog definition of xbit struct type, (b) illustration of the clock waveform implied by the member variables, value and t_offset.

In this case, both the input data and clock are xbit types and their events are happening within the same time step. Since SystemVerilog simulator does not guarantee a particular order for the events occurring within the same time step, it is possible for the clock event to occur before the data event does, even though its t_offset is smaller. It implies that a conventional sampler model listed in Fig. 2(b) may capture a wrong value of 0 rather than 1.

To prevent such synchronization failure, it is necessary to wait until all the input events within a time step have occurred and finally determine the output value according to their t_offset values. An improved sampler model listed in Fig. 3 illustrates how this can be done. It uses a *clocking* construct with *#1step* delay expression supported in SystemVerilog, which allows one to examine the inputs after all of their possible events have been triggered for the current time step. This sampler model essentially adds a unit-step delay between the input and output, but it is rather natural since all physical samplers have delays.

The second example is the expression of continuous-time analog signals. Although SystemVerilog allows a real type variable to be passed across the module port boundary, a series of its value change events basically expresses a piecewise-constant (PWC) waveform, as illustrated in Fig. 4(a). However, unless the events are sufficiently finely-spaced in time, a PWC waveform cannot accurately express the original signal. In comparison, a piecewise-linear (PWL) representation illustrated in Fig. 4(b) can reduce the error with the same number of events. A new struct type named xpwl supplements the signal's sampled value with the current slope and the time instant of the slope change event, as illustrated in Fig. 5.

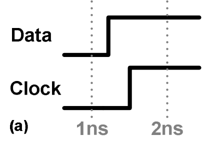
Yet, such PWL representation cannot accurately express certain signals with abrupt slopes. An example is the output of a RC loop filter in a second-order charge-pump PLL, as shown in Fig. 6. When the charge pump (CP) switches its current on or off, the voltage across the series resistor can change in steps. Moreover, the current pulse width can be short, especially when the PLL is close to a lock. Since the expression of a pulse needs at least two PWL events, the narrow pulse width would require again a fine time step.

It is noteworthy that the entire waveform of the loop filter output need not be modeled accurately in order to model the behavior of the overall PLL. The loop filter output is connected to a voltage-controlled oscillator (VCO), which updates its clock phase according to the time integral value of the control input. Therefore, it is only the integral value of the loop filter output that needs to be modeled accurately. A struct type xreal was defined to supplement the loop filter output signal with the integral value information as illustrated in Fig. 7. The explicit notation of the input's integral value also makes the VCO model simple. The VCO model can simply accumulate the integral values and multiplies it with a gain factor to derive the output phase.

III. Event-Driven Simulation of Linear Systems

This section describes how to model linear systems such as filters and lossy communication channels using the proposed data supplementation technique so that they can be efficiently simulated in an event-driven simulator.

The basic problem of modeling a dynamical system in an event-driven way is that the output signal can continuously change without any change of the input. For instance, once the



```
module sampler(input clk, input data, output q);
  always @(posedge clk)
    q = data;
endmodule
```
(b)

Fig. 2. A synchronization issue with a sampler model using xbit: (a) when both the data and clock events occur within the same time step, (b) a conventional sampler model may not capture the correct data depending on the event order which is guaranteed to be consistent with t_offset values.

```
module sampler( input xbit clk, input xbit data,
                output xbit q);
  always @(posedge ck.clk.value) begin
     if(data_pre == ck.data.value ||
        ck.data.err < ck.clk.err)
        q = data_pre;
     else
        q = ck.data.value;
     data_pre = ck.data.value;
  end

  always  #(1) tick <= ~tick;

  clocking ck @(tick);
     default input #1step;
     input clk, data;
  endclocking
endmodule
```

Fig. 3. An improved sampler model that resolves the synchronization issue.

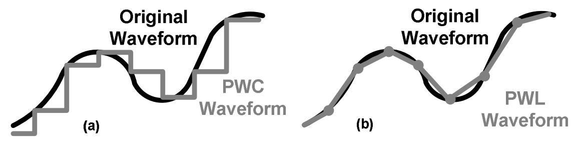


Fig. 4. Expressing continuous-time analog waveforms: (a) piecewise-constant (PWC) representation, (b) piecewise-linear (PWL) representation.

```
typedef struct {
   bit value;
   real slope;
   real t_offset;
} xpwl;
```
(a)

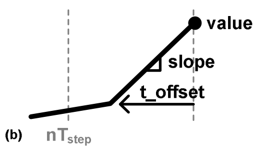


Fig. 5. Expressing PWL waveforms: (a) definition of xpwl struct type, (b) illustrated definitions of the member variables: value, slope, and t_offset.

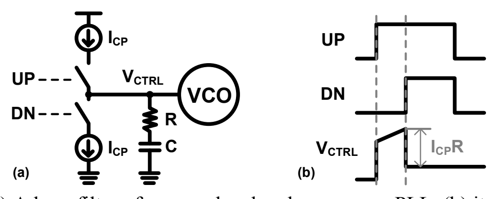


Fig. 6.(a) A loop filter of a second-order charge pump PLL, (b) its waveform.

```
typedef struct {
   bit value;
   real int_value;
} xreal;
```
(a)

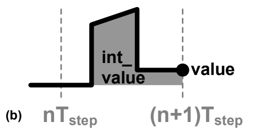


Fig. 7. Expressing the PLL loop filter output: (a) definition of xreal struct type, (b) illustrated definitions of the member variables: value and int_value.

input to a first-order low-pass filter changes in a step (i.e. a single event), its output exponentially decays towards the final value, of which expression requires multiple events even with the aforementioned xpwl or xreal type representations.

In case of a linear system, one way to mitigate this is to express the output waveform $y(t)$ as a weighted sum of complex-exponential terms:

$$y(t) = c_0 + c_1 e^{p_1 t} + c_2 e^{p_2 t} + ... + c_n e^{p_n t} \quad (1)$$

where c_0, c_1, …, c_n are real-valued coefficients while p_1, p_2, ..., p_n are complex-valued. In fact, p_1, p_2, ..., p_n are the system's eigenvalues (i.e. poles) and are fixed throughout the simulation for a given linear system. On the other hand, the coefficients c_0, c_1, …, c_n can change with the system's initial condition and with the input.

This way of expressing the linear system's output is effective since the coefficients c_0, c_1, …, c_n need to be updated only once when the input change event occurs. In other words, the model behavior is entirely event-driven. Also, the collection of these coefficients can be compactly expressed as a single struct-type (xexp) variable in SystemVerilog.

In case that the input signal a PWL waveform (i.e. xpwl type), all that is necessary is to modify the canonical expression for the output waveform to:

$$y(t) = c_0 + c_1 e^{p_1 t} + c_2 e^{p_2 t} + ... + c_n e^{p_n t} + c_{n+1} \cdot t \quad (2)$$

which is the general expression of a linear system's output in response to a ramp input, since a PWL waveform can be decomposed into a series of linear ramps, each initiated by a input change event.

IV. Experimental Results

The newly defined data types with supplemental information are applied in modeling a PLL, a lossy channel, and a high-speed link system to evaluate their effectiveness.

The first example is a second-order PLL, of which block diagram is illustrated in Fig. 8(a). All the digital signals such as clocks and up/down signals from the phase-frequency detector (PFD) are xbit's while the loop filter output is xreal. Fig. 9 compares the jitter histograms measured by the proposed PLL model with data supplementation and the conventional model that only relies on the time step to express accurate timings. The measured simulation speeds and errors in rms jitter values are plotted in Fig. 10. The models are simulated with MentorGraphics ModelSim simulator on a linux-based PC with AMD Phenom II X4 945 processor and 4GB memory. As expected, with a coarse time step of 10ps (1/100 of the clock period), the accuracy of the conventional model is very poor and approaches to a reasonable value only when the time step is less than or equal to 1ps (1/1000 of the clock period). Note that in SystemVerilog, the size of the time step can be adjusted only in logarithmic scale. On the other hand, the proposed model runs at the speed of 120,000 cycles/sec with a coarse time step of 100ps, and still has the better accuracy than the conventional model running at 1ps time step, which runs 50 times slower.

Fig. 8(b) plots the simulated jitter transfer functions of the proposed PLL compared with the theoretical ones [6]. For PLL configurations with different bandwidths and damping factors, the simulated results match closely with those expected in theories, validating the correctness of the models.

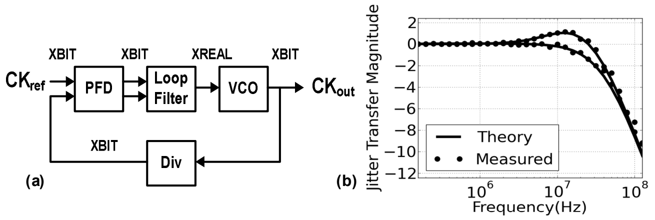


Fig. 8. A second order PLL (a) model and (b) the simulated vs. theoretical jitter transfer functions of the PLL. The bandwidth is 40MHz and damping factors are 1 for lower line and 0.1 for upper line.

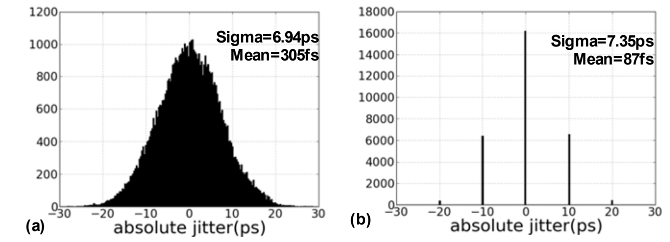


Fig. 9. Simulated jitter histogram of a 2GHz PLL: (a) the proposed model with 10ps time step, (b) the conventional model with 10ps time step.

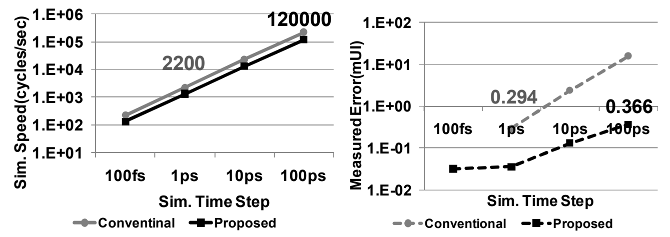


Fig. 10. Comparisons of the (a) simulation speeds and (b) standard deviation of output phase in response to step phase input (dashed line) when modeling a 2GHz PLL.

The second example is a lossy transmission line. A common approach is to sample the channel's impulse response and model its behavior based on a digital FIR filter model. However, for a lossy transmission line with long memory (i.e. long propagation delay), the impulse response itself can be very long and sampling it in a fine time step can incur significant simulation costs. The simulation speed of a FIR filter model would degrade with the length of the impulse response measured in sample periods. Furthermore, the situation is aggravated when one wants to model the data jitter effects and uses even finer time steps model them accurately.

Our proposed channel model takes an xpwl-type input and produces an xexp-type output. As discussed in Section III, the channel behavior can be simulated completely in an event-driven way. The eigenvalues of the channel is estimated from the measured S-parameter model using the *rationalfit* function available from Matlab. The coefficients of the complex exponential terms of the output signal are updated only once for each input change event. In addition, the output signal at a particular time instant is computed only when it is needed (e.g. when the receiver samples the signal triggered by the clock event) in order to save computational costs.

Fig. 11 compares the simulation speeds and accuracies of the proposed xexp-based model and the conventional FIR-based model for a channel with 2.9-dB loss at 1-GHz. The impulse response is 3ns long, implying that an FIR filter model with 1ps time step would have 3000 taps. In comparison, the xexp-based model required only 136 poles to achieve the similar accuracy with this FIR filter model. Furthermore, the proposed model ran about 600 times faster.

The third example is a serial link system depicted in Fig. 12. It consists of a second order PLL, a driver with pre-emphasis equalization, a channel, and a binary-controlled CDR. The input and output of channel are of xpwl and xexp types, respectively. Fig. 13 shows the eye diagram at the receiver side and the power spectral densities of the transmit and receive clocks. Fig. 14 compares the speed performances of the proposed and conventional models. Both models use the xexp-based channel model with 70 poles. The rms jitter of the CDR's recovered clock is measured to compare the accuracies. The results show that the conventional model requires a time step as fine as 100fs to achieve the similar accuracy with the proposed model running at 100ps time step. Therefore, the proposed model runs about 80 times faster than the conventional model with the similar accuracy.

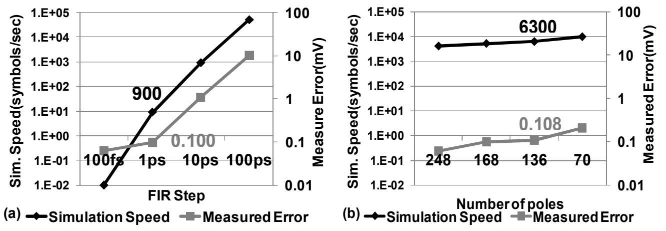


Fig. 11. Comparison of simulation speeds (black lines) and measured error (grey lines) of channel with (a) conventional model (b) proposed model.

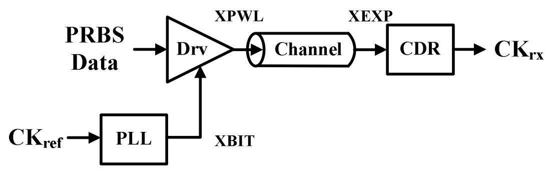


Fig. 12. Prototype serial link system. The bandwidth and damping factor of the PLL are 4MHz and 1 respectively, and CDR's bang-bang phase step is 1/50UI.Channel loss is -2.9dB at 1GHz.

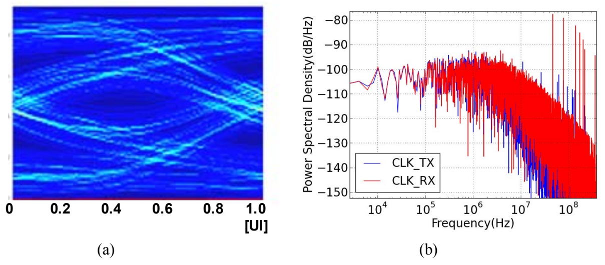


Fig. 13. Simulation results of a serial link: (a) the eye diagram at receiver side and (b) the phase noise spectrums of the transmit and receive clocks.

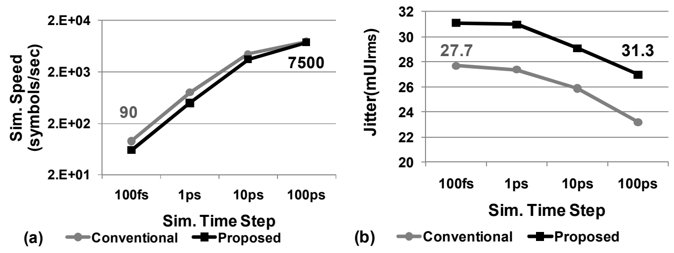


Fig. 14. Comparison of (a) Simulation speeds and (b) noise of recovered clock (dashed line) with conventional model (grey) and proposed model.

V. Conclusions

This paper described a way of simulating mixed-signal systems on an event-driven simulator efficiently and accurately based on data supplementation. Additional information is annotated on the signal either to improve the accuracy in expressing analog behaviors or to enable event-driven simulation of dynamical systems. We demonstrated the effectiveness of this approach for a variety of cases: expressing a clock or logic event with infinite time resolution (xbit), expressing a continuous-time, analog signal in piecewise-linear form (xpwl), expressing a PLL's loop filter response with emphasis on the integral value (xreal), and finally expressing a linear system's response with high precision in an event-driven fashion (xexp). For the examples listed, the achieved simulation speeds were 50 to 600 times faster than the conventional approaches with the equivalent accuracies. It should be noted that the proposed data supplementation technique is not limited to the demonstrated cases and can be extended to modeling other mixed-signal systems such as data converters and RF transceivers.

VI. Acknowledgments

This work has been funded by Synerchip, Inc., Sunnyvale, CA, U.S.A. and the CAD tool licenses have been supported by IC Design Education Center (IDEC) in Korea.

References

[1] K. S. Kundert, "Predicting the Phase Noise and Jitter of PLL-Based Frequency Synthesizers", in *Phase-Locking in High-Performance Systems: From Devices to Architectures*. Piscataway, NJ: IEEE Press, 2003, pp. 46–69
[2] M. van Ierssel, et al., "Event-Driven Modeling of CDR Jitter Induced by Power-Supply Noise, Finite Decision-Circuit Bandwidth, and Channel ISI", *IEEE Trans. Circuits and Systems I*, pp. 1306-1315, May 2008.
[3] R. B. Staszewski, et al., "Event-driven Simulation and Modeling of Phase Noise of an RF Oscillator", *IEEE Trans. Circuits and Systems I*, pp. 723-733, April 2005.
[4] A. Demir, et al., "Behavioral Simulation Techniques for Phase/delay-locked Systems", in *Proc. IEEE Custom Integrated Circuits Conference (CICC)*, pp. 453-456, May 1994.
[5] M. H. Perrott, "Fast and Accurate Behavioral Simulation of Fractional-N Frequency Synthesizers and other PLL/DLL Circuits", in *Proc. Design Automation Conference (DAC)*, pp. 498-503, Jun. 2002.
[6] F. Gardner, "Charge-Pump Phase-Lock Loops", *IEEE Trans. on Communications*, pp. 1849-1858, Nov. 1980.