Current Source Modeling in the Presence of Body Bias

Saket Gupta and Sachin S. Sapatnekar
University of Minnesota, Minneapolis, MN 55455, USA.
{saket, sachin}@umn.edu

***Abstract*— With the increasing use of adaptive body biases in high-performance designs, it has become necessary to build timing models that can include these effects. State-of-the-art timing tools use current source models (CSMs), which have proven to be fast and accurate. However, a straightforward extension of CSMs to incorporate multiple body biases results in unreasonably large characterization tables for each cell. We propose a new approach to compactly capture body bias effects within a mainstream CSM framework. Our approach features a table reduction method for compact storage, and a fast and novel waveform sensitivity method for timing evaluation. On a 45nm technology, we demonstrate high accuracy, with worst-case errors of under 5% in both slew and delay as compared to HSPICE. We show a speedup of over five orders of magnitude over HSPICE and almost 70× over conventional CSMs.**

I. INTRODUCTION

Variational effects in nanometer-scale technologies have motivated the need for presilicon and postsilicon approaches to addressing performance variability. One such approach that affords great adaptability and versatility is the idea of using adaptive body biases (ABBs) to compensate for performance variability. Realistically, ABB is applied at coarse levels of granularity, e.g., by biasing individual n-wells and/or p-wells, each of which contains a number of transistors.

The idea of body biasing is to deliberately apply a voltage to the body terminal of a transistor. Forward body bias (FBB) effectively reduces the transistor threshold voltage and speeds up the device, at the cost of increased leakage, while reverse body bias (RBB) achieves the opposite effect on speed and leakage. ABB involves the use of FBB or RBB to help dies recover from variations, and may be applied dynamically to tighten the distribution of the dies with maximum operational frequency, while simultaneously meeting the maximum leakage power [1], [2]. The effective use of ABB requires the ability to precharacterize the timing response of the circuit to the applied body bias, and this is the subject of this paper.

Traditional standard cell modeling approaches represent the delay and output slew as nonlinear functions of the input slew and output load capacitance [3]. With the onset of significant interconnect resistance, the notion of effective capacitance was introduced [4]. However, these approaches model the input as a saturated ramp with piecewise constant slope. Ideally, a ideal timing model should not assume a waveform shape; moreover, it should allow greater versatility in modeling a range of loads, beyond capacitances. These features are enabled by a class of models that are commonly termed as current source models (CSMs), which represent the cell as a controlled source, and provide fast and accurate timing estimates.

Blade [5] is a CSM approach that represents the cell as a voltage-controlled current source (VCCS) with an internal capacitance and a time-shifted input waveform driving an arbitrary load. A lookup table, indexed by the input voltage, V_{in}, and the output voltage, V_{out}, models the VCCS current, I_{out}. This was further refined in [6]–[11].

For ABB-based designs, it is necessary to evaluate the cell and circuit delays at various values of v_{bp}, the PMOS body bias, and v_{bn}, the NMOS body bias. Since ABB is applied at the granularity of a well, we assume that all PMOS transistors in a cell have the same body bias value, v_{bp}, and all NMOS devices are biased at v_{bn}.

This research was supported in part by the SRC under contract 2007-TJ-1572 and by the NSF under award CCR-0541367.

The obvious extensions to existing CSMs to handle body biases are rather inefficient. In principle, the body terminal of a device can be considered to be another port, and the cell can be accordingly characterized by creating a look-up table. However, this increases the amount of storage as well as the characterization time significantly over the zero body bias case. For instance, for 10 values each of v_{bp} and v_{bn}, the table for each cell becomes 100× larger. The need to access a larger lookup table may also result in a concomitant increase in the simulation runtime of CSM macromodels.

We develop a framework for incorporating effects of body bias into the CSM with a very small storage and runtime overhead, while maintaining high levels of accuracy. Our mathematical framework consists of two key steps. First, we intelligently adapt an existing scheme to enable the compact storage of look-up tables for the sensitivities of CSM components to body bias, over the range of allowable values of the body bias. Our second and more key contribution is to develop a novel waveform sensitivity model for evaluating the impact of the applied body bias, which provides accurate waveforms at the output of the cell under any body bias, with minimal computation. The essential idea of this approach is that since body bias constitutes a small perturbation to the nominal waveform, it should be possible to determine the perturbed waveform cheaply by determining and storing parameters that compute its shift from the nominal waveform. We develop a theoretically sound scheme for characterizing this perturbation and computing it efficiently.

The rest of the paper is organized as follows. Section II presents the development of sensitivity models for CSM components to handle body bias. Section III presents our algorithms for CSM table reduction, and is followed by a description of our method for fast output waveform evaluation in Section IV. Section V presents experimental results on a set of library cells in a 45nm technology.

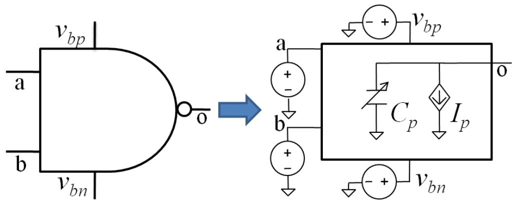


Fig. 1. Example of a CSM: the output port is modeled as a nonlinear VCCS dependent on all input port voltages, in parallel with a nonlinear capacitance.

II. CSM SENSITIVITY MODEL DEVELOPMENT

The CSM is a gate-level black-box abstraction of a cell in a library, with the same input and output ports as the original cell. Our CSM structure, shown in Figure 1, is of the type proposed in [5], and is augmented to model nonlinearities as in [8]. Specifically, output port p is replaced by a nonlinear VCCS, I_p, in parallel with a nonlinear capacitance, C_p. The VCCS model enables the CSM to be load-independent, and permits it to handle an arbitrary electrical waveform at its inputs. The CSM is characterized in terms of the value of I_p and the charge, Q_p, stored on the capacitor, C_p. The variables, I_p and Q_p, are functions of all input and output port voltages, and are determined by characterizing the cell at various port voltages and body bias combinations as follows:

$$I_p = F(V_i, V_o, v_{bp}, v_{bn}) \quad (1)$$
$$Q_p = G(V_i, V_o, v_{bp}, v_{bn}) \quad (2)$$

The parameters I_p and Q_p are modeled using the functions F and G, respectively, and V_i and V_o are, respectively, the voltages at the transitioning input and output ports of the cell. For a cell, I_p characterization involves DC simulations over multiple combinations of DC values of (V_i, V_o), while Q_p is characterized through a set of transient simulations [8]. The presentation of our model is targeted to the more widely-used scenario of single-input switching for gates with a single output, though the idea can easily be extended to multiple input switching (MIS) and multioutput gates, leveraging current work on CSMs on these topics [8]–[10].

As mentioned earlier, in order to capture the sensitivity of CSM parameters to the applied body bias (v_{bp}, v_{bn}), in principle, the circuit could be characterized over a set S of all possible (v_{bp}, v_{bn}) points, treating body terminals as input ports. Since the allowable values of the applied body biases change in discrete steps, the cardinality of this set is finite but large, and the corresponding characterization would be expensive and computationally intensive, even as a precharacterization step that is to be performed once for a technology. Moreover, table storage requirements multiply significantly over the current characterization procedure at zero body bias.

However, observing that the functions F and G depend *much more* weakly on v_{bn} and v_{bp} as compared to V_i or V_o, a simpler model can be utilized to save on this computation. We construct a polynomial approximation for the variation of I_p and Q_p with respect to (v_{bp}, v_{bn}); our simulations show that a first-order polynomial yields an average of 2% relative error with respect to HSPICE [12], evaluated over all (v_{bp}, v_{bn}) points, for all (V_i, V_o) points. The CSM is now represented using the equations:

$$I_p = I_p^Z \cdot (1 + a_I(V_i, V_o)v_{bp} + b_I(V_i, V_o)v_{bn}) \quad (3)$$
$$Q_p = Q_p^Z \cdot (1 + a_Q(V_i, V_o)v_{bp} + b_Q(V_i, V_o)v_{bn}) \quad (4)$$

where $I_p^Z = F(V_i, V_o, 0, 0)$, $Q_p^Z = G(V_i, V_o, 0, 0)$, and a_I, b_I, a_Q, and b_Q correspond to the sensitivity of the function to the corresponding body bias. These parameters are characterized at a discrete set of (V_i, V_o) values and stored in a lookup table.

The characterization of I_p and Q_p using Equations (3) and (4) can now be carried out using a minimum of three simulations at each (V_i, V_o), since it is a linear model; however, additional redundancy is preferable to account for the small nonlinearities.

For notational simplicity, we will define the following functions:

$$L_I(v_{bp}, v_{bn}) = (1 + a_I(V_i, V_o)v_{bp} + b_I(V_i, V_o)v_{bn}) \quad (5)$$
$$L_Q(v_{bp}, v_{bn}) = (1 + a_Q(V_i, V_o)v_{bp} + b_Q(V_i, V_o)v_{bn}) \quad (6)$$

III. Compact CSM Storage

A. Table size reduction for conventional CSMs

We apply the method in [13] to create compact lookup tables for I_p and Q_p for the zero body bias case, i.e., I_p^Z and Q_p^Z in Equations (3) and (4), with controlled loss of accuracy. For general values of the body bias, we must also store a_I, b_I, a_Q, and b_Q at each value of (V_i, V_o): as we will see, for these parameters, a direct extension of the method in [13] does not yield satisfactory results.

We first overview the procedure in [13]. This method begins with an $n \times n$ table of characterized points, indexed by variables x and y in the horizontal and vertical directions, respectively. The idea behind table size reduction is to store a subset of all these points and to interpolate the rest: a point can be removed from the table if its value can be accurately interpolated. For instance, consider the rectangle bounded by points $(x_1, y_1), (x_2, y_1), (x_2, y_2)$, and (x_1, y_2): a point (x, y) within this rectangle can be dropped if the interpolation error in its value, using these points lies within a specified bound. Instead of an expensive enumeration, the work in [13] presents a dynamic programming method for reducing a two-dimensional $n \times n$ table. The objective of the algorithm is to create a smaller $m \times m$ table, where m is prespecified, while minimizing the total error corresponding to the points that are dropped from the table. The procedure begins by constructing an initial 2×2 table corresponding to the points at the four corners of the table. Next, this table is expanded to include additional entries using the idea of h-hops.

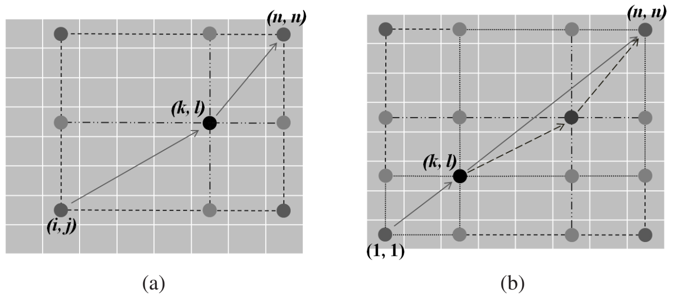


Fig. 2. (a) A 1-hop solution from (i, j) to (n, n), through an intermediate point, (k, l). (b) A 2-hop solution from $(1, 1)$ to (n, n) through an intermediate point, (k, l) uses a previously computed optimal 1-hop solution from (k, l) to (n, n).

In the initial step, we consider all rectangles originating at a point (i, j) at the southwest corner, extending to any point (k, l) at the northeast corner. We compute the error metric over the rectangle, corresponding to the case where only the points at the four corners of the rectangle are stored, and all internal points are dropped. The error metric is the sum of the interpolation errors for all points within and on the perimeter of the rectangle. Each such rectangle corresponds to an optimal substructure for dynamic programming: the optimal solution will be composed from some (but not all) such substructures.

Next, we define a 1-hop operation. We optimize the region bounded by point (i, j) to the southwest and (n, n) to the northeast by finding an optimal point (k, l) within this region. Here, optimality is defined as follows: the point (k, l) divides the region into four subregions, as shown in Figure 2(a), and over all candidate (k, l) points, the optimal point minimizes the total error summed up over these four subregions. Since the error over each rectangle was calculated in the initial step, this step involves enumerating all candidate (k, l) points, and summing up the previously calculated error over the rectangles in constant time for each such point. We refer to this as a 1-hop, indicating that for each (i, j), the table "hops" over a single point, corresponding to the optimal (k, l), on the way to (n, n). The associated optimal error encountered is the 1-hop error for (i, j).

In general, an h-hop from (i, j) to (n, n) finds a point (k, l) such that the error from (i, j) to (k, l), plus the $(h-1)$-hop error from (k, l) to (n, n), is minimized over all candidate points (k, l). To obtain an $m \times m$ table, the procedure stops after $m-1$ hops, and the optimal m-hop from $(1,1)$ to (n,n) provides the compact table. Figure 2(b) shows an example of a 2-hop solution from $P(1,1)$ to $P(n,n)$; if the algorithm were to stop here, it would result in a 4×4 compacted table. The computational complexity of this algorithm is $O(m \cdot n^4)$, but as m is typically small, this remains tractable.

B. Modifications for sensitivity tables

As stated earlier, the above approach works well for characterizing I_p and Q_p, where neighboring entries have similar magnitudes. However, in case of the sensitivity parameters, a_I, b_I, a_Q, and b_Q, there can be large differences in the values of neighboring parameters. This is illustrated in Figure 3, which shows the values of $b_I = \partial I_p/\partial v_{bn}$ and $a_Q = \partial Q_p/\partial v_{bp}$ for an inverter cell[1], where large "outliers" (i.e., values of large magnitude) are seen in a few places. For such data, it can easily be shown that the approach in [13] is poorly compressed, i.e., the interpolation errors in the reduced table are large. Such errors are demonstrated to be easily visible in the output response, where they appear as "spikes" in the CSM-based waveform that do not exist in the corresponding HSPICE waveform.

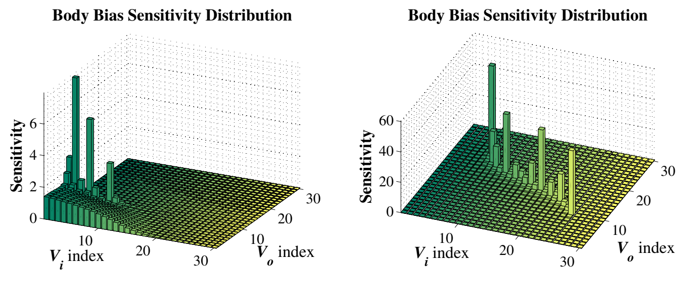


Fig. 3. The CSM sensitivity parameter distribution for (a) $\partial I_p/\partial v_{bn}$ and (b) $\partial Q_p/\partial v_{bp}$ as functions of (V_i, V_o).

We propose a simple method for avoiding these problems, based on the observation that for these sensitivity parameters, such outliers are few in number and have relatively large magnitudes. We therefore store the outliers separately. A typical distribution is shown in Figure 3, and it can be observed that the number of outliers is quite small compared to the total number of data points.

In order to store the outliers separately, given the set of all points, we find the mean and variance over all entries. Any entry that is over k variances from the mean is found to be an outlier; in practice, we find $k = 2$ to be an adequate value. The removed entry at table location (x, y) is then replaced by a dummy point, the error contribution (to the total error) from which is zero. The modified table is then compactly stored using the algorithm in Section III-A.

When a table entry is requested, we first determine whether the accessed point is an outlier: if so, we fetch it from the outlier list; else, we find it using the compressed look-up table.

IV. SENSITIVITY USAGE IN MACROMODEL SOLVER

A. Solving the macromodel

To solve the case of a gate driving an interconnect, including cases that involve coupled lines and crosstalk, it is enough to consider the situation where a gate drives a π-model, since standard techniques such as the O'Brien-Savarino approach [14] may be used to reduce an arbitrary interconnect load to a π-model at the driving point. Once the waveform at the driving point node V_o is obtained, evaluating the waveform at any sink node in the RC network involves solving a linear system using standard model order reduction methods.

[1]Similar behavior is seen for $a_I = \partial I_p/\partial v_{bp}$ and $b_Q = \partial Q_p/\partial v_{bn}$.

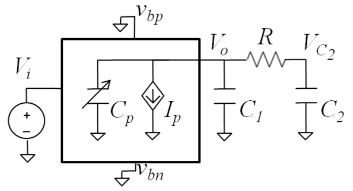


Fig. 4. A CSM for a gate, under zero body bias, driving a π load.

Therefore, we analyze the case of a gate output driving a π-load in the absence of body bias, as shown in Figure 4. Finding the output voltage waveform involves solving the equation:

$$I_p + I_{Q_p} = I_{C_1} + I_{C_2} \tag{7}$$
$$\text{where } I_{Q_p} = \frac{dQ_p}{dt}\ ;\ I_{C_1} = C_1\frac{dV_o}{dt}\ ;\ I_{C_2} = C_2\frac{dV_{C_2}}{dt}$$

Equation (7) is a nonlinear differential equation in V_o, and the input voltage, $V_i(t)$, is known. This equation can be solved using routine circuit simulation methods. We apply the Backward Euler formula to Q_p, V_o and V_{C_2} with a time step h, going from time n to time $n+1$ (the superscript $n+1$ is dropped for notational simplicity) to get:

$$Q_p = Q_p^n + hI_{Q_p} \tag{8}$$
$$C_1V_o = C_1V_o^n + hI_{C_1} \tag{9}$$
$$C_2V_{C_2} = C_2V_{C_2}^n + hI_{C_2} \tag{10}$$

Moreover, $V_{C_2} = V_o - RI_{C_2}$. Substituting V_{C_2} from this in (10):

$$I_{C_2} = \frac{C_2(V_o - V_{C_2}^n)}{h + RC_2} \tag{11}$$

Similarly, we obtain the values of I_{Q_p} from (8), of I_{C_1} from (9) and of I_{C_2} from (11), and substitute them in (7) to obtain:

$$I_p + \frac{Q_p - Q_p^n}{h} = \frac{C_1(V_o - V_o^n)}{h} + \frac{C_2(V_o - V_{C_2}^n)}{h + RC_2}$$

Solving this for V_o, we arrive at the following expressions:

$$V_o = \left[hC_2V_{C_2}^n + B(C_1V_o^n + hI_p + Q_p - Q_p^n)\right]/A \tag{12}$$
$$\text{where } A = (hC_1 + hC_2 + RC_1C_2) \tag{13}$$
$$B = (h + RC_2) \tag{14}$$

Obtaining V_o, we substitute I_{C_2} from (11) in (10) to obtain:

$$V_{C_2} = \left[hV_o + RC_2V_{C_2}^n\right]/B \tag{15}$$

B. Newton-Raphson Solver

To solve the nonlinear equation (12), we go through iterative Newton-Raphson linearization. In the $(k+1)^{\text{th}}$ iteration, we use the k^{th} iteration value, shown by the additional subscript k, to obtain:

$$V_o = V_{o,k} - \frac{AV_{o,k} - hC_2V_{C_2}^n - B(C_1V_o^n + hI_{p,k} + Q_{p,k} - Q_p^n)}{A - B(h\ \partial I_p/\partial V_o|_k + \partial Q_p/\partial V_o|_k)} \tag{16}$$

This computation is carried out by references to the look-up tables for I_p and Q_p, with the appropriate use of interpolation as necessary, and the use of finite differences to compute derivatives.

C. Formulation of waveform sensitivity model

The above Newton-Raphson solver forms the basis for a procedure for computing the waveform under any body bias condition. However, if the gate delay is to be evaluated under numerous body bias conditions, this entails multiple simulations of the *entire* output voltage waveform at each body bias value. Applications that require timing analysis at multiple body biases include [1], [2], [15], [16].

Intuitively, the repeated computation of full waveforms from scratch seems unnecessarily excessive, for two reasons. First, the application of body bias corresponds to a perturbation to a base case, such as the zero body bias case, and it should be possible to compute the waveform based on the zero body bias case, with some consideration of body bias sensitivities, much more cheaply than the above procedure. Second, in most cases, designers are interested not in the entire waveform, but specific properties of the gate output, such as its delay and output transition time. In this section, we demonstrate the efficient computation of such metrics under body bias without the need for numerous table look-up operations.

For various values of (v_{bn}, v_{bp}), the solution of the waveform under the framework of Equation (16) entails multiple accesses to the look-up tables for I_p and Q_p. The entries that are accessed in these tables change according to the applied body bias. However, since body bias is a small perturbation, in practice, the accessed entries in each table at each step of the algorithm are relatively close to each other, and can be viewed as perturbations to a nominal case.

Therefore, we propose to capture the output waveform for nonzero body bias case as a perturbation to the zero body bias case, i.e.,

$$V_o(t) \quad = V_o^Z(t) + \alpha(v_{bp}, v_{bn}, t) \cdot v_{bp} + \beta(v_{bp}, v_{bn}, t) \cdot v_{bn} \quad (17)$$

where $V_o^Z(t)$ represents the output waveform, $V_o(t)$, when zero body bias is applied, and $\alpha(v_{bp}, v_{bn}, t)$ and $\beta(v_{bp}, v_{bn}, t)$ are time-varying perturbation parameters that are precisely defined as:

$$\alpha(v_{bp}, v_{bn}, t) = \frac{\partial V_o(t)}{\partial v_{bp}} \;; \beta(v_{bp}, v_{bn}, t) = \frac{\partial V_o(t)}{\partial v_{bn}} \qquad (18)$$

The following result provides a precise formula for $\alpha(t)$ and $\beta(t)$. We first present the result (proved in the Appendix), and then discuss how the computational cost of its evaluation can be significantly reduced.

Theorem 1 *The waveform sensitivity parameters, $\alpha(v_{bp}, v_{bn}, t)$ and $\beta(v_{bp}, v_{bn}, t)$, are given by:*

$$\alpha(v_{bp}, v_{bn}, t) = \; N_\alpha / D \qquad (19)$$
$$\beta(v_{bp}, v_{bn}, t) = \; N_\beta / D \qquad (20)$$

$$N_\alpha = B\Bigg[\alpha^n C_1 + h a_I I_p^Z + a_Q Q_p^Z - a_Q^n Q_p^{Z,n} - \frac{\partial Q_p^{Z,n}}{\partial v_{bp}} L_Q^n(v_{bp}, v_{bn}) - Q_p^{Z,n}\frac{\partial a_Q^n}{\partial V_o^n}\alpha^n v_{bp} - Q_p^{Z,n}\frac{\partial b_Q^n}{\partial V_o^n}\alpha^n v_{bn}\Bigg] + hC_2\frac{\partial V_{C_2}^n}{\partial v_{bp}},$$

$$N_\beta = B\Bigg[\beta^n C_1 + h b_I I_p^Z + b_Q Q_p^Z - b_Q^n Q_p^{Z,n} - \frac{\partial Q_p^{Z,n}}{\partial v_{bn}} L_Q^n(v_{bp}, v_{bn}) - Q_p^{Z,n}\frac{\partial a_Q^n}{\partial V_o^n}\beta^n v_{bp} - Q_p^{Z,n}\frac{\partial b_Q^n}{\partial V_o^n}\beta^n v_{bn}\Bigg] + hC_2\frac{\partial V_{C_2}^n}{\partial v_{bn}},$$

$$D = A - B\Bigg[\left(hI_p^Z\frac{\partial a_I}{\partial V_o} + Q_p^Z\frac{\partial a_Q}{\partial V_o}\right)v_{bp} + h\frac{\partial I_p^Z}{\partial V_o}L_I(v_{bp}, v_{bn}) + \left(hI_p^Z\frac{\partial b_I}{\partial V_o} + Q_p^Z\frac{\partial b_Q}{\partial V_o}\right)v_{bn} + \frac{\partial Q_p^Z}{\partial v_{bp}}L_Q(v_{bp}, v_{bn})\Bigg]$$

where terms using the superscript n are understood to correspond to their values at the previous (n^{th}) time step, and the superscript Z refers to the case where $v_{bn} = v_{bp} = 0$.

Theorem 1 enables the efficient computation $V_o(t)$, at any body bias value, using a closed form expression, dependent only on the values of V_o at previous time steps and the values in the zero body bias waveform. As a result, the waveform at arbitrary body bias values can be reproduced if the values of $\alpha(t)$ and $\beta(t)$ are computed.

D. Simplified waveform sensitivity model

Further simplifications are possible. On investigating dependency of the output waveform on (v_{bp}, v_{bn}) and on $\alpha(v_{bp}, v_{bn}, t), \beta(v_{bp}, v_{bn}, t)$, we observe that:

1) The variation in $V_o(t)$ over (v_{bp}, v_{bn}) is nearly linear at each time point of the waveform. Empirically, this can be seen in Figure 5, which shows typical cases for the variation of $V_o(t)$ over (v_{bp}, v_{bn}) for various time points of simulation. This behavior is observed for multiple test cases, and indicates that $\alpha(v_{bp}, v_{bn}, t), \beta(v_{bp}, v_{bn}, t)$ are actually independent of the applied body bias, and are only dependent on t.
2) Figure 6 shows the variations in $\alpha(v_{bp}, v_{bn}, t)$ and $\beta(v_{bp}, v_{bn}, t)$ with (v_{bp}, v_{bn}). The magnitude of these variations were observed to be a maximum of 0.1 for all test cases. Since these parameters are further multiplied by v_{bp} or v_{bn} $\in [-0.3, 0.3]$ in (17), their effects on $V_o(t)$ are expected to be negligible. This is further validated in Section V.

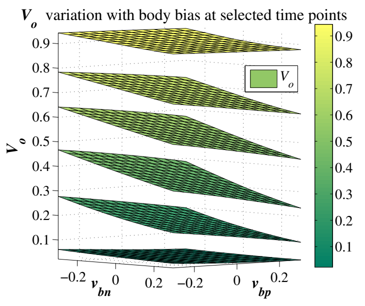


Fig. 5. Typical surface plots for V_o showing the linear nature of V_o variations with (v_{bp}, v_{bn}), with each surface corresponding to a randomly selected time point during the simulation.

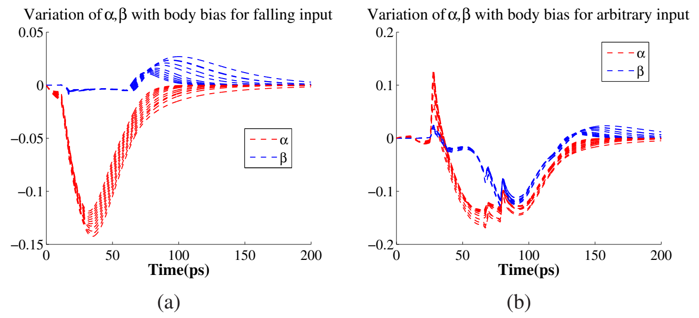


Fig. 6. Simulations showing the variation of $\alpha(t)$ and $\beta(t)$ at a range of body biases from the minimum to the maximum, including zero. Two such test cases are shown in Figure (a) and (b).

This leads to the following approximation, which provides accurate waveforms with very low errors, as demonstrated in Section V:

$$\alpha(v_{bp}, v_{bn}, t) \approx \alpha_0(t) = \alpha(v_{bp} = 0, v_{bn} = 0, t)$$
$$\beta(v_{bp}, v_{bn}, t) \approx \beta_0(t) = \beta(v_{bp} = 0, v_{bn} = 0, t) \qquad (21)$$

To summarize, evaluating the output at b body bias points each for v_{bp} and v_{bn} using an enumerative approach would solve for b^2 waveforms, involving the extensive use of lookup tables. In contrast, our approach reduces the solution to finding just three waveforms: one for the zero body bias, $V_o^Z(t)$, and one each for $\alpha_0(t)$ and $\beta_0(t)$. The net result is a massive savings in the computation. Thus, the steps involved in computing the waveform at any (v_{bp}, v_{bn}) are:

1) Apply (16) to generate the waveform $V_o^Z(t)$ at zero-body bias.
2) Compute and store $\alpha_0(t)$, $\beta_0(t)$ at every timestep from (19), (20), and (21).
3) Use the computed $\alpha_0(t)$, $\beta_0(t)$ in (17) to directly generate the waveform for any value of (v_{bp}, v_{bn}).

V. Experimental Results

A. Reduction in CSM sensitivity table size

We apply our table reduction algorithm for the sensitivity parameters, a_I, b_I, a_Q, and b_Q for a set of standard cells characterized using 45nm PTM [17], and demonstrate our results in Table I for a typical table, for a_I. Columns 2 through 4 show the number of entries in the reduced table using the original compression approach (Section III-A), and Columns 5 through 7 list the size of the reduced tables using our approach (Section III-B). These comparisons are shown for various bounds (2%, 5%, 10%) on the allowable error, and in each case, the optimal table size corresponds to the smallest $m \times m$ table that meets the error bound. In each case, $m = 30$ for the original table size, i.e., it has 900 entries. In each case, our approach yields much smaller tables than the prior approach. The last column of Table I shows the runtime of the algorithm for achieving reduced table sizes for the most computationally-intensive solution, where the 2% error bound must be satisfied. The runtimes are measured on a 3GHz Intel Core2Duo CPU, and correspond to the average for the a_I, a_Q, b_I, and b_Q sensitivity tables, and are very reasonable, especially considering that this characterization computation must be performed only once for a given library in a given technology.

It is easy to explain why the original algorithm of Section III-A does not lead to sufficient reduction in the table size. This can primarily be related to outliers: ignoring these points causes substantial errors at these points when interpolation is used to predict the values of missing entries; on the other hand, if these are included, the large jumps at these points can result in interpolation errors at nearby points that do not correspond to outliers. These errors can only be diminished by using reduced tables of larger sizes.

TABLE I
Results for sensitivity parameter table reduction for tables with original size = 900

Cell Type	Reduced Table Size with Error Bounds						Run Time
	Original approach (Section III-A)			Our approach (Section III-B)			
	2%	5%	10%	2%	5%	10%	
INV	529	484	324	225	169	100	115s
NAND2	576	484	289	196	144	81	110s
NOR2	900	784	576	324	256	169	168s
NAND3	625	529	256	169	144	81	104s
NOR3	841	729	484	289	225	144	167s
AOI21	576	529	484	196	169	100	114s
AOI22	529	484	361	225	169	81	117s

B. Results for our waveform sensitivity model

1) Accuracy and speedup: We now present the accuracy of and the speedup attained using our simplified waveform sensitivity (WS) model, as proposed in Section IV-D.

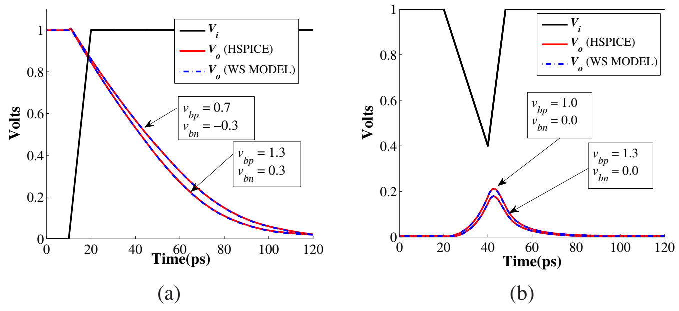


Fig. 7. (a) The result of our waveform sensitivity (WS) method for an Inverter, as compared with HSPICE, for several body bias values. (b) A similar set of waveforms for a NAND2, modeling a glitch.

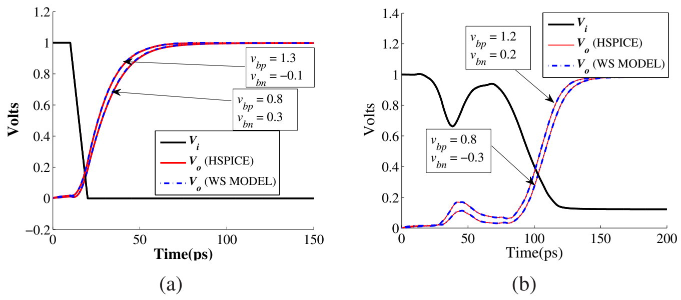


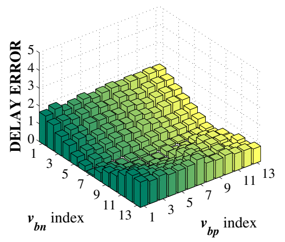
Fig. 8. The result of our waveform sensitivity (WS) method for a NAND3, as compared with HSPICE, for several body bias values. (a) shows the output for a monotone input waveform, while (b) shows a nonmonotone input.

Figures 7 and 8 compare representative waveforms as generated through HSPICE [12] and the simplified waveform sensitivity model, when the input waveform takes any arbitrary shape either due to glitches, noise or crosstalk. Each figure shows, for various body bias values, the response of a specific gate to an input waveform. The waveforms are coincident to the naked eye, so that our algorithm yields high accuracy. This also validates the idea that α, β can be assumed to be independent of (v_{bp}, v_{bn}), as proposed in Section IV-D.

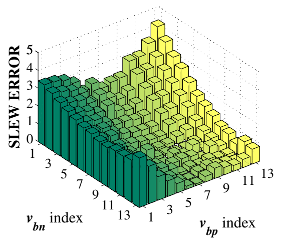
TABLE II
Speedups obtained by our Simplified Waveform Sensitivity (WS) Model over HSPICE and Newton-Raphson (NR) solver

Cell	WS Model Speedup	
	Over HSPICE	Over NR Solver
INV	8.9e4	65.36
NAND2	9.6e4	66.29
NOR2	9.2e4	69.23
NAND3	9.6e4	66.67
NOR3	8.9e4	72.15
AOI21	10.8e4	66.80
AOI22	10.0e4	69.36

Next, we evaluate the speedup of our method over HSPICE and over the Newton-Raphson solver (see section IV-B) that would be used in a simple extension of existing CSMs. To calculate the above speedup, we perform our tests with each circuit example under multiple combinations of the following parameters: multiple rise/fall waveforms (1ps–100ps input ramps, in steps of 10ps), various capacitance (0–15fF, in steps of 3fF) and resistance (2–10K, in steps of 2K) values for the π-load, and multiple body bias points (169 points with $(v_{bp}, v_{bn}) \in [-0.3, 0.3]$, in steps of 0.05V for each parameter). For each combination of these parameters, we calculate the runtimes using HSPICE, Newton-Raphson solver and our simplified waveform sensitivity model, and average these runtimes over all the test cases to arrive at speedup results. For the test cases, we perform transient simulations and report the speedups of our algorithm over HSPICE and over the Newton-Raphson in Table II. Expectedly, the speedup over HSPICE is large, and is found to be about five orders of magnitude. More interestingly, our model achieves an average speedup of 67.9× over the Newton-Raphson solver.



(a) Average percent delay error



(b) Average percent slew error

Fig. 9. The delay and slew errors of our waveform sensitivity model, relative to HSPICE, over 169 (v_{bp}, v_{bn}) points for a NAND2 cell.

2) Accurate delay and slew calculations: For the above test cases, we also tabulate the delay and slew values (for rise/fall waveforms) as obtained from our waveform sensitivity model and from HSPICE. We then obtain the average error between delay corresponding to the waveform sensitivity model as compared to that from HSPICE. Similar computations are performed for the slew error. Figure 9 shows the average delay and average slew error for a NAND2. Clearly, only a small error is incurred in both delay and slew, validating the use of our waveform sensitivity model in predicting the delay and slew over all (v_{bp}, v_{bn}) points. Similar observations were made for other standard cells and all of the errors found to be within 5%.

VI. Conclusion

A simple extension of existing CSM's to incorporate body bias effects in the CSM framework results in huge increase in library storage and solver runtime. We present a novel approach to incorporate body bias into current source models. We develop sensitivity model for capturing variations in CSM components with body bias, with compact storage for the resulting tables of these model parameters. We incorporate this sensitivity model into the mainstream CSM solver framework, and develop a new model for capturing waveform sensitivity with body bias, allowing us to compute waveforms at multiple body bias points with massive savings in computation. The results demonstrate the effectiveness of our compression scheme and the waveform sensitivity model in achieving HSPICE level accuracy with high speedup both over HSPICE and conventional CSM solvers.

References

[1] T. Kuroda, T. Fujita, S. Mita, T. Nagamatsu, S. Yoshioka, K. Suzuki, F. Sano, M. Norishima, M. Murota, M. Kako, M. Kinugawa, M. Kakumu, and T. Sakurai, "A 0.9-V, 150 MHz, 10-mW, 4 mm^2, 2-D discrete cosine transform core processor with variable threshold-voltage (VT) scheme," in *Proceedings of the IEEE International Solid-State Circuits Conference*, pp. 166–167, 1996.

[2] J. W. Tschanz, J. Kao, S. G. Narendra, R. Nair, D. Antoniadis, A. Chandrakasan, and V. De, "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage," *IEEE Journal of Solid-State Circuits*, vol. 37, pp. 1396–1402, November 2002.

[3] S. S. Sapatnekar, *Timing*. Boston, MA: Springer, 2004.

[4] F. Dartu, N. Menezes, and L. Pileggi, "Performance computation for precharacterized CMOS gates with RC loads," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 15, pp. 544–553, May 1996.

[5] J. F. Croix and D. F. Wong, "Blade and Razor: Cell and interconnect delay analysis using current-based models," in *Proceedings of the Design Automation Conference*, pp. 386–389, 2003.

[6] I. Keller, K. Tseng, and N. Verghese, "A robust cell-level crosstalk delay change analysis," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 147–154, 2004.

[7] P. Li and E. Acar, "A waveform independent gate model for accurate timing analysis," in *Proceedings of the International Conference on Computer Design*, pp. 363–365, 2005.

[8] C. Amin, C. Kashyap, N. Menezes, K. Killpack, and E. Chiprout, "A multi-port current source model for multiple-input switching effects in CMOS library cells," in *Proceedings of the Design Automation Conference*, pp. 247–252, 2006.

[9] N. Menezes, C. Kashyap, and C. Amin, "A "true" electrical cell model for timing, noise, and power grid verification," in *Proceedings of the Design Automation Conference*, pp. 462–467, 2008.

[10] B. Amelifard, S. Hatami, H. Fatemi, and M. Pedram, "A current source model for CMOS logic cells considering multiple input switching and stack effect," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 568–573, 2008.

[11] S. Raja, F. Varadi, M. Becer, and J. Geada, "Transistor level gate modeling for accurate and fast timing, noise, and power analysis," in *Proceedings of the Design Automation Conference*, pp. 456–461, 2008.

[12] http://www.synopsys.com/products/mixedsignal/hspice/hspice.html.

[13] J. F. Croix and D. F. Wong, "A fast and accurate technique to optimize characterization tables for logic synthesis," in *Proceedings of the Design Automation Conference*, pp. 337–340, 1997.

[14] P. R. O'Brien and T. L. Savarino, "Modeling the driving point characteristic of resistive interconnect for accurate delay estimation," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 512–515, 1989.

[15] J. Tschanz, N. Kim, S. Dighe, J. Howard, G. Ruhl, S. Vanga, S. Narendra, Y. Hoskote, H. Wilson, C. Lam, M. Shuman, C. Tokunaga, D. Somasekhar, S. Tang, D. Finan, T. Karnik, N. Borkar, N. Kurd, and V. De, "Adaptive frequency and biasing techniques for tolerance to dynamic temperature-voltage variations and aging," in *Proceedings of the IEEE International Solid-State Circuits Conference*, pp. 292–604, 2007.

[16] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "Body bias voltage computations for process and temperature compensation," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 16, pp. 249–262, March 2008.

[17] Predictive Technology Model. http://www.eas.asu.edu/∼ptm.

Appendix

Proof of Theorem 1

At each time step, combining the nonlinear equation (12) with (18), and A, B given by (13), (14), we can write $\alpha(t)$ as:

$$\alpha(t) = \frac{BC_1}{A}\frac{\partial V_o^n}{\partial v_{bp}} + \frac{hC_2}{A}\frac{\partial V_{C_2}^n}{\partial v_{bp}} + \frac{B}{A}\left(h\frac{\partial I_p}{\partial v_{bp}} + \frac{\partial Q_p}{\partial v_{bp}} - \frac{\partial Q_p^n}{\partial v_{bp}}\right) \quad (22)$$

Writing I_p, Q_p as linear functions of v_{bp}, v_{bn} from (3), (4), we get

$$\alpha(t) = \frac{BC_1}{A}\frac{\partial V_o^n}{\partial v_{bp}} + \frac{hC_2}{A}\frac{\partial V_{C_2}^n}{\partial v_{bp}} + \frac{B}{A}\Bigg[h\frac{\partial I_p^Z}{\partial V_o}\frac{\partial V_o}{\partial v_{bp}}L_I(v_{bp}, v_{bn}) +$$
$$hI_p^Z\left(a_I + \frac{\partial a_I}{\partial V_o}\frac{\partial V_o}{\partial v_{bp}}v_{bp} + \frac{\partial b_I}{\partial V_o}\frac{\partial V_o}{\partial v_{bp}}v_{bn}\right) + \frac{\partial Q_p^Z}{\partial V_o}\frac{\partial V_o}{\partial v_{bp}}L_Q(v_{bp}, v_{bn}) +$$
$$Q_p^Z\left(a_Q + \frac{\partial a_Q}{\partial V_o}\frac{\partial V_o}{\partial v_{bp}}v_{bp} + \frac{\partial b_Q}{\partial V_o}\frac{\partial V_o}{\partial v_{bp}}v_{bn}\right) - \frac{\partial Q_p^{Z,n}}{\partial V_o^n}\frac{\partial V_o^n}{v_{bp}}L_Q^n(v_{bp}, v_{bn}) -$$
$$Q_p^{Z,n}\left(a_Q^n + \frac{\partial a_Q^n}{\partial V_o^n}\frac{\partial V_o^n}{\partial v_{bp}}v_{bp} + \frac{\partial b_Q^n}{\partial V_o^n}\frac{\partial V_o^n}{\partial v_{bp}}v_{bn}\right)\Bigg]$$

where L_I and L_Q are defined as in Equations (5) and (6), respectively, and L_Q^n corresponds to the evaluation of L_Q at time step n. Further, $\partial V_{C_2}/\partial v_{bp}$ can be calculated using Equation (15).

Recognizing that $\partial V_o/\partial v_{bp} = \alpha$, and at time step n, $\alpha^n = \partial V_o^n/\partial v_{bp}$, collecting all terms multiplied by α, the result of Equation (19) follows immediately. The derivation of Equation (20) is analogous. Note that a_I, a_Q, b_I and b_Q are independent of body bias, being functions of (V_i, V_o) only, but appear as functions of (v_{bp}, v_{bn}) since V_o dynamically changes with body bias during simulation.