A Gate-Level Timing Model for SOI Circuits†

Mehrdad Shahriari and Farid N. Najm
Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario M5S-3G4

Abstract – Partially depleted floating-body (PDFB) SOI technology offers the potential of increased speed and lower power dissipation over traditional bulk CMOS. A key problem, however, related to the use of traditional design flows for new SOI designs is that the delay of logic gates built of PDFB SOI transistors varies according to the signal history. This presents a complication for doing timing analysis or simulation of logic circuits. In this paper, we have formulated a simulation model that allows one to track the changes in delay during a dynamic gate-level simulation of the PDFB SOI circuit. This is essential in order to properly account for the shift of transistor body voltage in SOI devices. The model captures the "state" of a logic gate via two delay "state variables" which represent the rise and fall delay of the logic gate. As the logic circuit is simulated, state variables become updated.

1. Introduction

Partially depleted floating-body (PDFB) SOI transistors [1] offer improved performance and lower power compared to CMOS [2, 3], but lead to complications in the design flow due to the fact that the transistor body is floating, and therefore has variable voltage [4, 5]. As a logic gate experiences a series of logic transitions, the body voltages of its transistors will change [6]. There are two types of changes that determine the dynamics of the body voltage, *fast* changes and *slow* changes [7]. The fast changes occur when a logic gate undergoes a transition. Due to capacitive coupling between the gate and drain nodes, and due to the transistor impact ionization current, the body voltage of an n-channel MOSFET in an inverter, for example, will experience a fast step up when the inverter input rises. It will experience a fast step down when the input falls. It can be shown that these fast body voltage changes due to logic transitions depend on the input rise/fall time, the output load capacitance and the body voltage values before the transition of the logic gate. When the gate is in steady state, the body voltage will drift slowly due to pn-junction currents. For the n-channel in an inverter, forward biased source junction currents wtill cause a slow change of body voltage [6, 7]. The time scales of these changes are very different. The fast changes happen over pico-seconds or nano-seconds, while the slow changes take milli-seconds [7]. A good way to visualize the process is to think of the body node as a charge tank. For a given signal waveform, charge is pumped into the tank and taken out of the tank, and the final voltage is a function of the signal history [6, 7]. Since transistor threshold voltage depends on the body potential, and since the delay of a logic gate depends on threshold voltage, then the delay of a logic gate becomes a function of the signal history [4, 6, 7].

In a logic gate, the body voltage values of all the transistors will determine the delay value (because they affect V_T, which affects the delay [6]). One can think of the various body voltages inside a logic gate as representing the *state* of the gate for delay computation purposes. However, building a gate-level timing model that tracks the separate body voltage values would lead to a very complex model, especially for complex gates, because one would have to track all the internal voltages of the gates as well, leading to a model that is close to a transistor-level simulator. To overcome this, based on our observation that logic gates with *different* body voltage values but the *same* delay value experience very similar delay variations in response to the same input signal stream, we propose to use the gate delay itself as a way to capture the gate *state*.

2. Inverter Model

We have developed a model for the basic static SOI inverter that maintains information on the rising delay (delay due to a rising input), and the falling delay (delay due to a falling input) of the inverter, as state variables. These values will be updated as the simulation proceeds, without maintaining any body voltage information. In the following, we describe the motivation and structure of the model. In section 2.3, we will describe how the model would be used inside a simulator. We will also verify the model against SPICE in section 2.4.

2.1. Motivation

The motivation behind our model is the following empirical observations. Different body voltage combinations inside the inverter can correspond to the same delay value. This can be seen from the contour plots in Fig. 1. Furthermore, if we start the inverter in two different body voltage combinations that correspond to the same delay, and we then apply in each case the same signal stream, the inverter delay and the way it changes over time will be approximately the same. This is shown in Fig. 2, which contains two groups of three curves each. The curves are so close that it is hard to tell them apart. In each

†This research is supported by the Semiconductor Research Corporation, under contract SRC 2000-TJ-755.

group, one curve starts from time 0, and corresponds to the inverter starting from a DC steady state and being clocked towards its AC steady state. The other two curves in each group correspond to the inverter starting in a different body voltage combination, with some corresponding delay, and then being simulated for the same 200MHz pulse sequence. Each of these two curves is then drawn on the plot starting from a carefully chosen initial time instant, as follows: given the initial starting delay value for each curve, we find the time when the first (DC) curve has that delay value, and we start each of the two curves from that time instant. This achieves a condition whereby the three curves correspond to three versions of the inverter that have the same initial delay but have different initial body voltage combinations. It is clear that, having the same delay value and in spite of the different body voltage combinations, the inverters have approximately the same delays under subsequent inputs. This behavior was found to apply in all cases, and is the basis for our model. Fig. 3 shows the percentage spread between final delays for simulation runs of a typical inverter that is started in different body voltage states for various initial delay values. The results shown are for a total of 216 SPICE runs. This test has been performed for all ranges of load and input slope for an inverter, NAND and NOR gates, with up to three inputs.

2.2. Model structure

The inverter model maintains *two state variables*, D_r & D_f, defined as the propagation delays due to a rising input and a falling input. For each, the model contains two main components and one auxilary component.

1. Δ-Delay tables: The first main component is the "Δ-Delay" tables, or simply the ΔD tables. This component captures the delay change due to a transition, and is shown in Fig. 4(a) for a rising input (a similar table gives ΔD for a falling input). The tables give the change in delay due to a rising or falling transition, given the delay value immediately before the transition. Several versions of each table are required, each corresponding to a certain output load (capacitance) and input rise/fall time combination. These tables are generated by starting from a DC steady state (one high and one low) and applying a sequence of fast pulses to the gate and measuring the delay values using SPICE.
2. Delay decay tables: The second main component is the "delay decay" tables set, and is shown in Fig. 4(b), for a rising input (again, another similar table exists for a falling input). This component captures the delay change due to staying at logic high or low, so that we have two decay curves for each state variable. Several tables are required, to account for all the delay and input slope combinations.
3. Mapping tables: The auxiliary component is a set of "mapping" tables, as shown in Fig. 4(c). This component captures the relation between delay values for different input slopes with a fixed load, and allows one to use delay values characterized for a certain input slope to compute delays due to another. Several tables are required, to account for the different load values.

2.3. Model usage

We now give a step-by-step description of how the model is used, in the general case when the input slopes in a waveform may be different, as shown in Fig. 5. We start from input state low, and right before the first transition (with rise time t_{r1}), we assume that the state variables D_{r0} and D_{f0} are known. The following three functions will denote the components of the model.

1. $\delta_{r/f}(D, t)$ is the Δ-Delay function, based on the Δ-Delay tables. The first argument is the delay before the transition and the second argument is the input rise/fall time. The result is ΔD due to that transition. The r/f subscript stands for rise or fall.
2. $\lambda_{r/f,\text{high/low}}(D, t, T)$ is the decay function, based on the decay tables. The first argument is initial value of delay. The second argument is the input rise/fall time, and the third argument is the decay time. The result is the final value of delay. The r/f subscript stands for rise and fall. The low/high subscript specifies if it is a decay to low or decay to high.
3. $\mu_{r/f}(D, t_s, t_d)$ is the mapping function, based on the mapping tables. The first argument is the delay value for a given slope, and the second argument is the rise/fall time associated with the first argument. The third argument is the rise/fall time value for which we want to compute the delay. The r/f stands for for rise or fall.

At point A, right before the first transition we have:

$$D_{r,A} \triangleq D_r = D_{r0}$$

$$D_{f,A} \triangleq D_f = D_{f0}$$

At point B, after the first transition, we compute the new delay values using the ΔD tables, as follows:

$$D_r = D_{r,A} + \delta_r(D_{r,A}, t_{r1}) \triangleq D_{r,B}$$

$$D_f = D_{f,A} \triangleq D_{f,B}$$

At point C, right before the second transition, we update the state variables using the decay tables, as follows:

$$D_r = \mu_r(\lambda_{r,\text{high}}(D_{r,B}, t_{r1}, t_{\text{high}}), t_{r1}, t_{r2}) \triangleq D_{r,C}$$

$$D_f = \mu_f(\lambda_{f,\text{high}}(D_{f,B}, t_{r1}, t_{\text{high}}), t_{r1}, t_{f2}) \triangleq D_{f,C}$$

The reason for this peculiar use of the mapping function is that it is much cheaper to create the other tables using the same input slope throughout a waveform, than to vary the slope combinations within a waveform. The mapping function is a cheap mechanism of using such tables in the general case.

At point D, we use this ΔD tables again to compute the state variables after the transition, as follows:

$$D_r = D_{r,C} \triangleq D_{r,D}$$

$$D_f = D_{f,C} + \delta_f(D_{f,C}, t_{f2}) \triangleq D_{f,D}$$

At point E, right before the third transition, we update the state variables using the decay tables, then we map this value to the slope t_{r3}, as follows:

$$D_r = \mu_r(\lambda_{r,\text{low}}(D_{r,D}, t_{r2}, t_{\text{low}}), t_{r2}, t_{r3}) \triangleq D_{r,E}$$

$$D_f = \mu_f(\lambda_{f,\text{low}}(D_{f,D}, t_{r2}, t_{\text{low}}), t_{r2}, t_{f3}) \triangleq D_{f,E}$$

At point F, applying the same update as used for point B, we have:

$$D_r = D_{r,E} + \delta_r(D_{r,E}, t_{r1}) \triangleq D_{r,F}$$

$$D_f = D_{f,E} \triangleq D_{f,F}$$

2.4. Experimental results

In order to show the accuracy of this model, we used an inverter and constructed a table of $(t_{\text{high}}, t_{\text{low}})$ combinations representing different duty cycle values and, for each combination, we simulated the inverter in SPICE and compared our delay prediction at the end of the cycle to that obtained from SPICE. A typical percentage error histogram is shown in Fig. 6, where t_{high} and t_{low} are from the set 1ns, 10ns, 100ns, 1us, 10us, 100us, 1ms, therefore we have 49 combination of input pulses. The errors are all under 5%, which is very good agreement, even by typical accuracy of bulk CMOS gate timing models.

It is important to investigate whether the error accumulates over time. To do so, we have constructed and simulated different tables of $(t_{\text{high}}, t_{\text{low}})$ combinations, as follows, with the results shown in Fig. 7(a). A 3×3 table denotes a matrix of all combinations of values of t_{high} and t_{low} from the set {1ns, 10ns, 100ns}. Thus a 3×3 table simulation in Fig. 7(a) consists of a simulation of the following sequence of cycles: $(t_{\text{high}}, t_{\text{low}}) = \{(1,1), (1,10), (1,100), (10,1), (10,10), (10,100), (100,1), (100,10), (100,100)\}$, all values in nsec. The 4x4 table corresponds to combinations of values of t_{high} and t_{low} from the set {1ns, 10ns, 100ns, 1000ns}, and similarly for the other tables. Notice that this set of pulses covers fast pulses as well as pulses that take the inverter near to its DC steady state in each half cycle. The error values are all small, again, justifying the use of this model. The errors also do not seem to accumulate. In the two cases of the 3×3 and the 4×4 tables, where Fig. 7(a) seems to show increasing error, we repeated the simulation a number of times, and the results are shown in Fig. 7(b), indicating that the error does not accumulate in these cases as well.

3. Extension to Other Gates

This modeling approach can, in principle, be used for other gates. So far, the model has been tested for two input NAND gates and a typical model accuracy is shown on Fig. 8. However, it may not be practical to simply increase the number of decay tables according to the set of input states (2^n). Instead, we are working on an approximate technique by which a fixed number of decay tables can be used.

4. Conclusion and Future Work

In this paper we have presented a model that can be used for timing simulation/analysis of FBPD SOI logic gates, we showed empirical results for an inverter and a 2-input NAND gate. The concept can be extended for other types of static gates as well as gates with larger number of inputs. Future work will include testing the model for gates with more number of inputs as well as AOI gates.

References

[1] K. Bernstein and N.J. Rohrer, *SOI Circuits Design Concepts*, Boston, MA: Kluwer Academic Publishers, 2000.

[2] G. Shahidi et.al, "Mainstreaming of the SOI Technology," *IEEE International SOI Conference, Proceedings*, pp. 1–4, Oct. 1999.

[3] A. O. Adan et.al, "SOI as a Mainstream IC Technology," *IEEE International SOI Conference, Proceedings*, pp. 9–12, Oct. 1998.

[4] R. Puri and C.T. Chuang, "SOI Digital Circuits: Design Issues," *Thirteenth International Conference on VLSI Design*, pp. 474–479, Jan. 2000.

[5] M. Canada et.al, "A 580MHz RISC Microprocessor in SOI," *IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, pp. 430–431, Feb. 1999.

[6] M. M. Pelella et.al, "Hysteresis in Floating-Body PD/SOI CMOS Circuits," *International Symposium on VLSI Technology Systems and Applications*, pp. 278–281, 1999.

[7] K. L. Shepard and Dae-Jin Kim, "Body-voltage estimation in digital PD-SOI circuits and its application to static timing analysis," *IEEE/ACM Intl. Conference on Computer-Aided Design*, pp. 531–538, 1999.

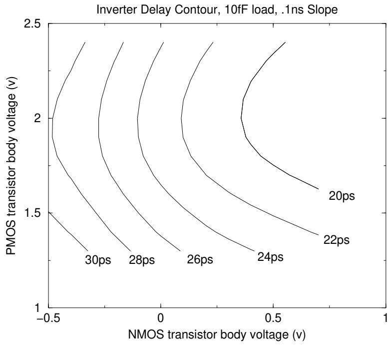


Figure 1. Equal-delay lines in the body voltage space for an inverter.

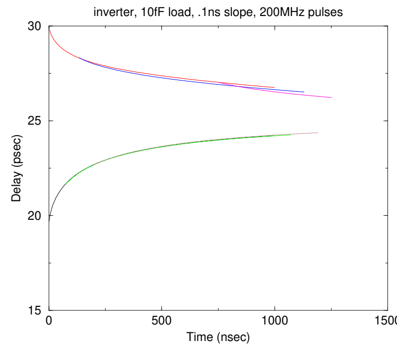


Figure 2. Inverter delay change, from different initial body voltages.

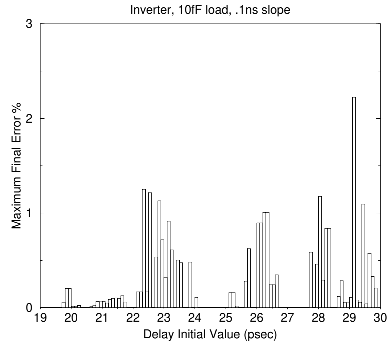
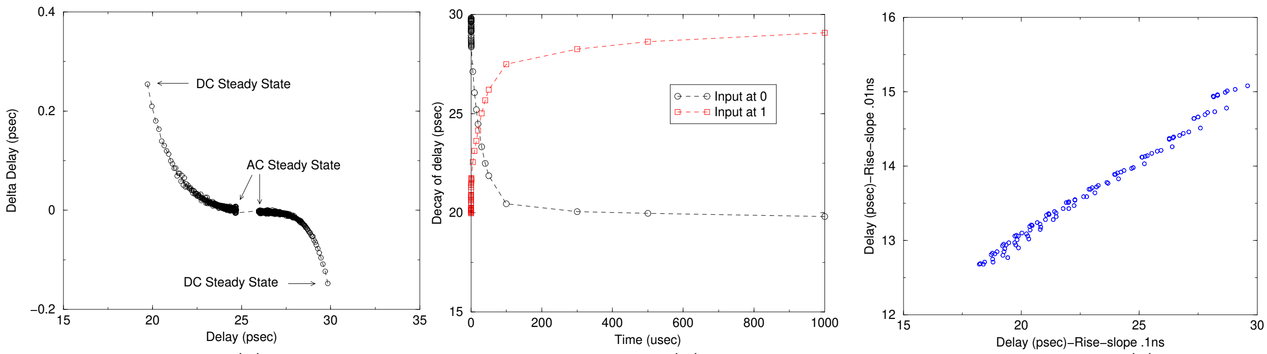


Figure 3. Percentage spread in final delay, for same initial delay.



(a) **(b)** **(c)**

Figure 4. Rise time ΔD table, load 10fF, input slope .1ns (a), rise time decay table, load 10fF, input slope .1ns (b), Mapping table for rise time, load 10fF (c).

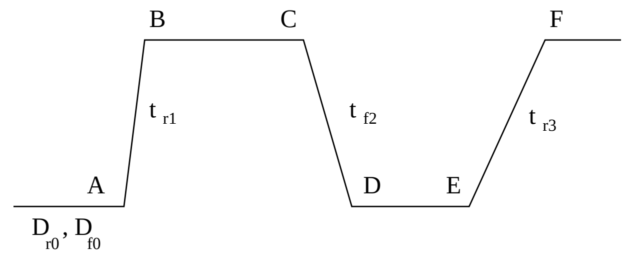


Figure 5. Example of an input pulse with different slopes.

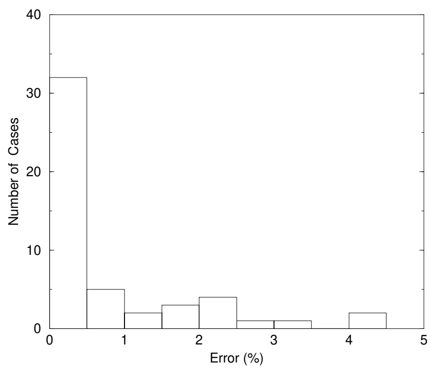
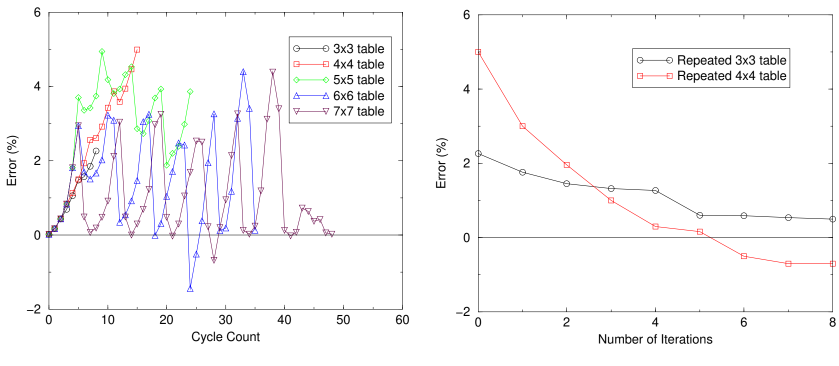


Figure 6. Percentage error/cycle relative to SPICE.



(a) **(b)**

Figure 7. Percent errors for different sets of $(t_{\text{high}}, t_{\text{low}})$ combinations (a), with repeated application of the same run for two cases (b).

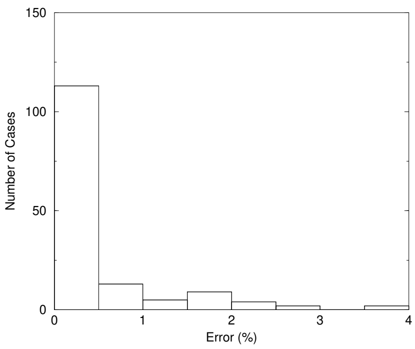


Figure 8. Error relative to SPICE, after one cycle.