Modeling the Effects of Temporal Proximity of Input Transitions on Gate Propagation Delay and Transition Time

V. Chandramouli and Karem A. Sakallah

CSE-TR-262-95

October 10, 1995

THE UNIVERSITY OF MICHIGAN

Computer Science and Engineering Division
Department of Electrical Engineering and Computer Science
Ann Arbor, Michigan 48109-2122
USA

Modeling the Effects of Temporal Proximity of Input Transitions on Gate Propagation Delay and Transition Time

V. Chandramouli and Karem A. Sakallah

Advanced Computer Architecture Laboratory
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, Michigan 48109-2122

October 10, 1995

Abstract

While delay modeling of gates with a single switching input has received a lot of attention, the case of multiple inputs switching in close temporal proximity is just beginning to be addressed in the literature. The effect of proximity of input transitions can be significant on the delay and output transition time. The few attempts that have addressed this issue are based on a series-parallel transistor collapsing method that reduces the multi-input gate to an inverter. This limits the technique to CMOS technology. Moreover, none of them discuss the appropriate choice of voltage thresholds to measure delay for a multi-input gate. In this paper, we first present a method for the choice of voltage thresholds for a multi-input gate that ensures a positive value of delay for any combination of input transition times and the temporal separations among them. We next introduce a dual-input proximity model for the case when only two inputs of the gate are switching. We then propose a simple algorithm for calculating the delay and output transition time that makes repeated use of the dual-input proximity model and that does not collapse the gate into an equivalent inverter. Comparison with simulation results shows that our method performs quite well in practice. Before concluding the paper we also show the close relationship between the inertial delay of a gate and the proximity of input transitions.

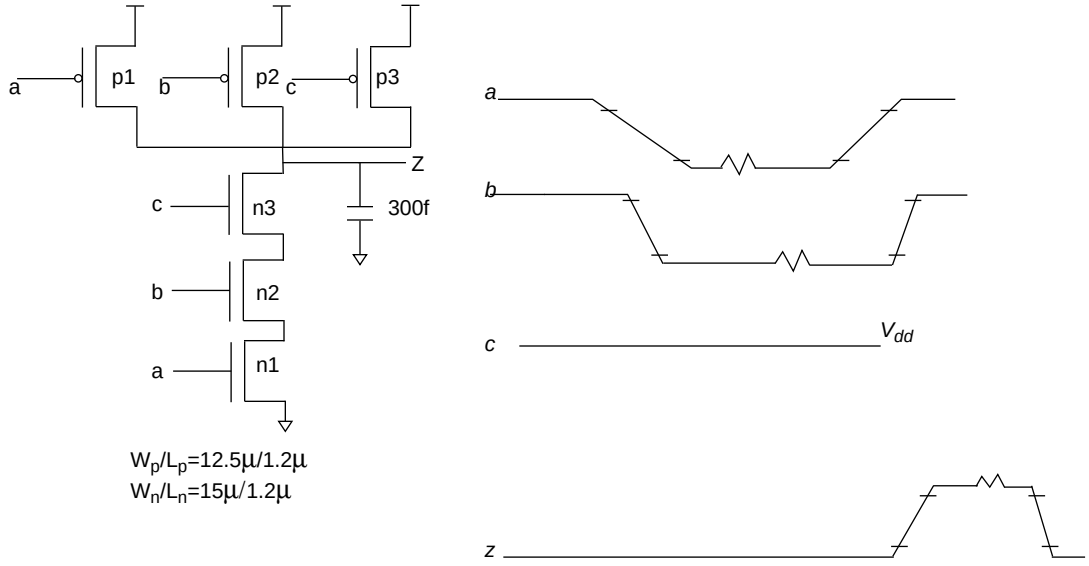
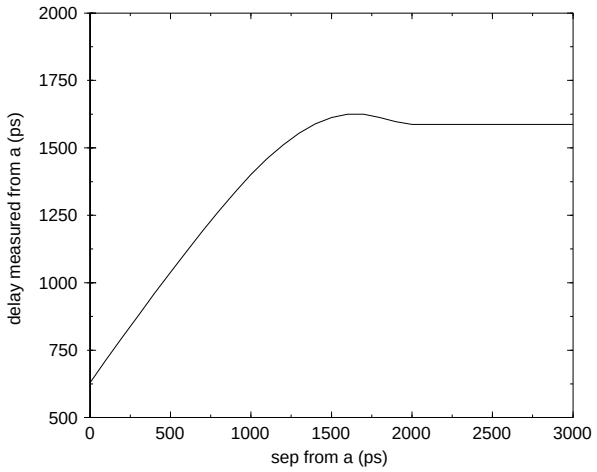


Figure 1-1: A 3-input NAND gate with temporally close transitions on its inputs

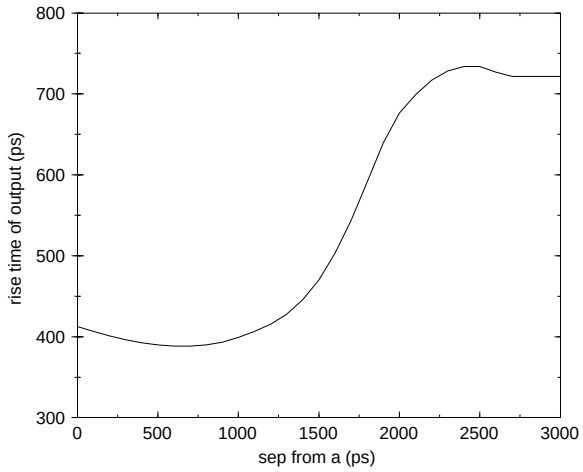
1 Introduction

Development of accurate delay models continues to be a critical need for high-performance VLSI applications. The combined effect of submicron feature sizes and larger die areas are forcing a reassessment of the conventional models for gate and interconnect delays. The earliest gate delay models accounted for such effects as load capacitance and transistor sizes [2]. More recently, the dependence of delay on the finite transition times of digital signals has been recognized by several researchers [6], [10], [14], [16] and incorporated in commercial delay calculators [18]. In this paper, we address the dependence of gate delay on the *temporal proximity* of input transitions. This effect was identified by a number of researchers [8], [13] and represents a form of state-dependency [5], [17]. To date, however, the modeling approaches proposed to capture proximity effects are incomplete, inaccurate, or specific to particular design styles. The proximity model we introduce in this paper attempts to remedy these shortcomings.

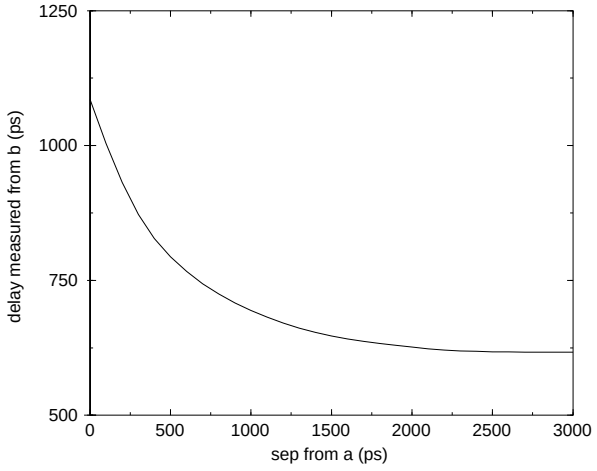
To illustrate the effect of transition proximity on gate delay consider the three-input CMOS NAND gate shown in Figure 1-1 and assume that inputs *a* and *b* experience, respectively, slow and fast falling transitions while input *c* is stable at V_{dd}. Figure 1-2(a) depicts the variation of gate delay as a function of the temporal separation between the transitions on *a* and *b*. For sufficiently large separations, the transition on *b* is blocked by the controlling value on *a* (logic 0) and does not affect gate delay. As the separation decreases, however, the p-transistor connected to *b* starts to conduct and provides another current path from V_{dd} to the output. As a result, the output rises faster and the effective gate delay is reduced. As the figure shows, the reduction in gate delay due to this proximity phenomenon can be significant. A similar effect can be observed for the rise time on the output (Figure 1-2(b)). Consider next the case when inputs *a* and *b* experience rising transitions while input *c* is stable at V_{dd}. As Figure 1-2(c) and (d) show, gate delay and output fall time become decreasing functions of separation. This can be readily explained by examining the behavior of the n-transistors in the pull-down stack.



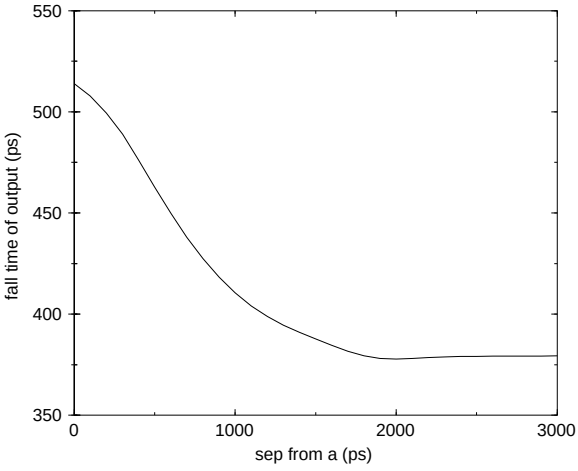
(a) Variation of delay for falling inputs



(b) Variation of output rise time for falling inputs



(c) Variation of delay for rising inputs



(d) Variation of output fall time for rising inputs

Figure 1-2: Variation of delay and output transition times as a function of separation be-

It should be clear from this simple example that the variation in delay due to temporal proximity can be significant and should be modeled if accurate delay estimations are sought. However, while the case of single-input switching has received a lot of attention, the proximity effect is just starting to be addressed in the literature. Many attempts to model the delay of multi-input gates assume only one input is switching [3], [11], [15] and thus do not take temporal proximity of input transitions into account. We now review the research that does consider multiple switching inputs. In [8], an equivalent waveform is found from the ones that are switching and the multi-input gate is collapsed into an inverter by series parallel reduction of the transistors. The justifications for deriving the equivalent waveform are not clearly stated. In addition, the output loading and input transition times are not taken into account while collapsing the transistors. As noted in [13], this can lead to large errors and an

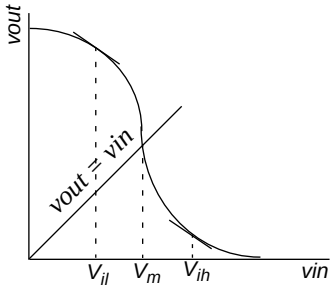
attempt has been made in that work to take loading and input transition times into account while reducing the gate. Their method reduces to that of [8] when the inputs switch together and their transition times are small. This method also finds an equivalent input waveform which is then applied to the inverter derived from the multi-input gate and is principally geared towards calculating the peak supply current. Percentage error for delay and output transition times are not given. However, all these techniques give significant errors when we measure delay and output transition time. Moreover, these techniques assume a CMOS implementation of the gate since they are based on combining series-parallel transistors. In this paper we propose a novel technique for computing the delay and output transition times that does not collapse a multi-input gate to an inverter. While our technique can be applied to any technology, we illustrate it with CMOS technology in this paper.

The rest of this paper is organized as follows. In Section 2, we describe how to choose appropriate voltage thresholds for multi-input gates to ensure positive delays. In Section 3, we formulate the delay and output transition time for a two-input gate and propose a temporal proximity model for the gate. In Section 4, we develop the proximity model for a multi-input gate using this dual-input proximity model. The experimental validation of our model for a three-input NAND gate is presented in Section 5. We base our comparisons on circuit simulations performed using HSPICE [12]. We digress briefly, in Section 6, to show the relationship between inertial delay and the proximity effect, and conclude the paper in Section 7, by summarizing our contribution and indicating future work.

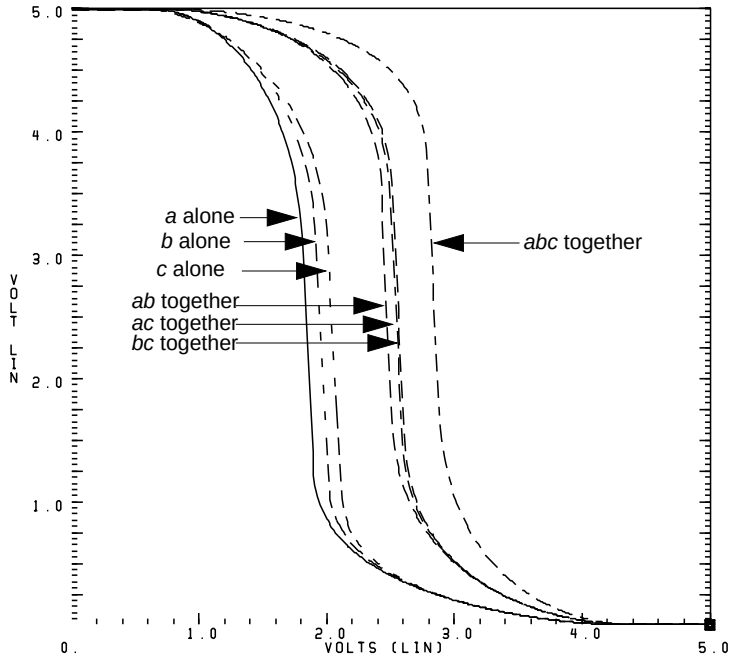
2 Defining delay thresholds

Delay is measured from the time when an input signal crosses a certain threshold (henceforth called the input threshold) to the time when the output signal crosses another threshold (henceforth called the output threshold). It is important to choose these thresholds carefully so that the delay is always positive. A typical Voltage Transfer Curve (VTC) for an inverter is shown in Figure 2-1(a). V_{il} and V_{ih} denote the points where the slope of the VTC is -1 [7]. V_m, also known as the switching threshold of the gate, denotes the point where the $Vout = Vin$ line intersects the VTC (usually close to $V_{dd}/2$). Traditionally, the input and output thresholds are chosen to be $V_{dd}/2$. However, it can be easily shown that a choice of input threshold above (below) V_m for rising (falling) input can give rise to negative delays for very slow inputs [4]. Thus, $V_{dd}/2$ is not a robust choice, as it could be slightly lower or higher than V_m. Choosing V_m instead is not useful either because the delay approaches zero in the limit as the input is made arbitrarily slow. Moreover, it is difficult to pinpoint V_m precisely because this is the point at which the gain of the inverter is maximum. Hence, some researchers have used V_{il} (V_{ih}) for the input threshold and V_{ih} (V_{il}) for the output threshold in case of rising (falling) inputs [10]. This definition of delay always gives a monotonically increasing delay value with increasing input transition time. These two thresholds also provide a logical choice for measuring input and output transition times.

However, in the case of multi-input gates with many inputs switching in close temporal proximity, it is not clear how to determine appropriate thresholds for delay measurement. Rather than a single VTC, an *n*-input gate can have $2^n - 1$ VTCs corresponding to all possible combinations of stable and switching inputs. Figure 2-1(b) shows the VTCs of the gate in Figure 1-1 obtained by circuit simulation. The V_{il}, V_{ih} and V_m of each curve are listed in Figure 2-1(c). The curve for the case when *a* is switched alone and the curve for the case when all of them switch together are the two extreme cases of this family of curves. To illustrate the difficulty in choosing appropriate thresholds for measuring delay, consider the case when the three input signals *a*, *b*, and *c* are rising together with the same transition times. Clearly we would choose the thresholds of the VTC corresponding to the case when all of them switch together, which is the last column of the table. Now, suppose that the temporal separation between input *a* and the other two inputs is increased. To ensure positive delay, we must shift from the VTC corresponding to all of them switching at the same time to the VTC when input *a* alone is switched, which corresponds to the first



(a) Inverter VTC



(b) VTC of a 3-input NAND gate

Thresholds	Switching Inputs						
	a	***b***	***c***	***a, b***	***a, c***	***b, c***	***a, b, c***
V_{il}	1.25V	1.35V	1.44V	2.05V	2.12V	2.15V	2.51V
V_m	1.87V	1.97V	2.06V	2.47V	2.54V	2.56V	2.83V
V_{ih}	2.16V	2.24V	2.33V	2.95V	2.99V	3.01V	3.37V

(c) Possible Switching Thresholds for 3-input NAND

Figure 2-1: VTC for a three input NAND gate

column of the table. This is explained as follows: when input *a* is separated sufficiently apart from *b* and *c*, the delay will approach the delay when *a* is switched alone, since *b* and *c* are stable at the non-controlling value. Therefore, the output will behave as if *a* were switching alone. In order to ensure positive delay, even for very slow rise time on *a*, we must ensure that $V_{il} < V_m$ of the gate. In this case, V_m would be obtained from the first column since it is *a* that is causing the output to switch. This property is not satisfied by V_{il} obtained from the last column and the final delay value (when *a* arrives very late) could become negative if we continue using thresholds from this column to measure delay. The exact point when we move from one VTC to the other is not clear. This situation is even more complicated for gates with fan-in greater than three.

To ensure that negative delays never arise and to avoid moving from one VTC to another depending on separation of inputs, we base our delay measurement on the minimum V_{il} and the maximum V_{ih} from all the VTCs. This will guarantee that $V_{il} < V_m < V_{ih}$ for V_m chosen from any VTC and will therefore ensure positive delay, no matter how many inputs are switch-

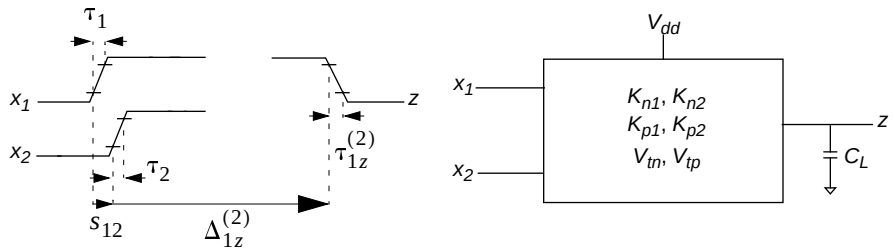


Figure 3-1: A black box model of a two-input CMOS gate

ing and how separated they are with respect to each other. In case of a NAND gate, the V_{il} chosen would be from the input closest to the ground and V_{ih} would be from the VTC corresponding to all inputs switching at the same time. For the case of NOR gates, V_{il} would be chosen from the VTC corresponding to all inputs switching at the same time and V_{ih} chosen from the input closest to the power rail. Thus, in our example NAND gate, V_{il} would be 1.25V and V_{ih} would be 3.37V. These thresholds were used in generating the curves in Figure 1-2.

Having determined a suitable choice of thresholds, we next show how to formulate the delay and output transition time as functions of the several parameters mentioned in the Introduction, for a two-input gate. We also show how the temporal parameters can be separated from the rest and use this as a basis for developing the proximity model for a two-input gate.

3 A dual-input temporal proximity model

In this section, we derive a proximity macromodel for the delay and output transition time of two-input gates. Starting from a complete enumeration of all waveform and circuit parameters that can affect delay, we show how these macromodels can be expressed as three argument functions. This derivation is based on the application of dimensional analysis and the invocation of reasonable simplifying assumptions.

Consider the black-box model of a two-input CMOS gate shown in Figure 3-1. Let $\Delta_{iz}^{(k)}$ and $\tau_{iz}^{(k)}$ denote the delay and output transition time, respectively, when k inputs are switching in close temporal proximity. The delay is assumed to be measured relative to input x_i, the *reference* input. We measure separation between two inputs by using V_{ih} for falling inputs and V_{il} for rising inputs. In its most general form, the delay and output transition time functions for this gate can be written as:

$$\Delta_{iz}^{(2)} = D_i^{(2)}(\tau_i, \tau_j, s_{ij}, C_L, V_{dd}, K_{ni}, K_{nj}, K_{pi}, K_{pj}, V_{tn}, V_{tp}) \tag{3.1}$$

$$\tau_{iz}^{(2)} = T_i^{(2)}(\tau_i, \tau_j, s_{ij}, C_L, V_{dd}, K_{ni}, K_{nj}, K_{pi}, K_{pj}, V_{tn}, V_{tp}) \tag{3.2}$$

where, $i, j = 1, 2$ such that $i \neq j$, $\Delta_{iz}^{(2)}$is the delay measured from input x_i when both the inputs are switching, $\tau_{iz}^{(2)}$ is the transition time of the output z when both inputs are switching, τ_i and τ_j are the transition times of the corresponding inputs, s_{ij} is the separation between the two inputs measured from input x_i, C_L is the total load capacitance (including interconnect capaci-

tance), V_{dd} is the supply voltage, K_{ni}, K_{nj}, K_{pi} and K_{pj} are the strengths[1] of the n- and p-transistors of the corresponding inputs, and V_{tn} and V_{tp} are the threshold voltages of the n- and p-transistors. We can make some reasonable assumptions which simplify these functions somewhat. In most designs, all n-transistors are of the same size as are all p-transistors. Therefore, the individual transistor strengths can be replaced by one parameter for the n-transistors and one for the p-transistors. We can further simplify the model by separating the effects of temporal and non-temporal parameters. We note that delay and transition time functions, when input x_i alone is switching, can be written as:

$$\Delta_{iz}^{(1)} = D_i^{(1)}(\tau_i, C_L, V_{dd}, K_n, K_p, V_{tn}, V_{tp}) \tag{3.3}$$

$$\tau_{iz}^{(1)} = T_i^{(1)}(\tau_i, C_L, V_{dd}, K_n, K_p, V_{tn}, V_{tp}) \tag{3.4}$$

Dimensional analysis has been shown to be a powerful tool in reducing the number of parameters in a macromodel [9]. Using this technique (3.3) and (3.4) can be written as follows:

$$\frac{\Delta_{iz}^{(1)}}{\tau_i} = D_i^{(1)}\left(\frac{C_L}{K_n V_{dd}\tau_i}, \frac{K_p}{K_n}, \frac{V_{tn}}{V_{dd}}, \frac{V_{tp}}{V_{dd}}\right) \tag{3.5}$$

$$\frac{\tau_{iz}^{(1)}}{\tau_i} = T_i^{(1)}\left(\frac{C_L}{K_n V_{dd}\tau_i}, \frac{K_p}{K_n}, \frac{V_{tn}}{V_{dd}}, \frac{V_{tp}}{V_{dd}}\right) \tag{3.6}$$

In a cell-based design environment and for a given process, the ratios of the pullup and pulldown transistors is fixed and the designer has no control over the threshold voltages. These ratios can then be absorbed into the function and the resulting functional forms are [9]:

$$\frac{\Delta_{iz}^{(1)}}{\tau_i} = D_i^{(1)}\left(\frac{C_L}{K_n V_{dd}\tau_i}\right) \tag{3.7}$$

$$\frac{\tau_{iz}^{(1)}}{\tau_i} = T_i^{1}\left(\frac{C_L}{K_n V_{dd}\tau_i}\right) \tag{3.8}$$

Note that the functions $D_i^{(1)}$ and $T_i^{(1)}$ in (3.5), (3.7) and (3.6), (3.8) are not the same. However, to reduce clutter we use the same symbols. This will hold true throughout the rest of the paper.

Equations (3.7) and (3.8) capture the effect of the non-temporal parameters. By conjecturing that proximity delay is a perturbation of the delay due to a single input transition, and using (3.3) and (3.4), we can write our original delay and transition time equations as:

$$\Delta_{iz}^{(2)} = D_i^{(2)}(D_i^{(1)}(\tau_i, C_L, V_{dd}, K_n, K_p, V_{tn}, V_{tp}), \tau_i, \tau_j, s_{ij}) \tag{3.9}$$

---

1. $K = \frac{1}{2}\mu C_{ox}\frac{W}{L}$ where μ is the carrier mobility, C_{ox} is the capacitance per unit area and W, L are the transistor width and length, respectively.

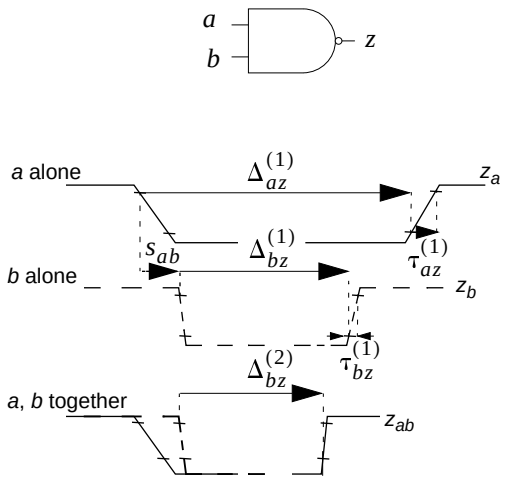


Figure 3-2: Determining the causing input

$$\tau_{iz}^{(2)} = T_i^{(2)}(T_i^{(1)}(\tau_i, C_L, V_{dd}, K_n, K_p, V_{tn}, V_{tp}), \tau_i, \tau_j, s_{ij}) \tag{3.10}$$

These two equations have 4 parameters each. By using dimensional analysis, we can reduce them to the following forms [9]:

$$\frac{\Delta_{iz}^{(2)}}{\Delta_{iz}^{(1)}} = D_i^{(2)}\left(\frac{\tau_i}{\Delta_{iz}^{(1)}}, \frac{\tau_j}{\Delta_{iz}^{(1)}}, \frac{s_{ij}}{\Delta_{iz}^{(1)}}\right) \tag{3.11}$$

$$\frac{\tau_{iz}^{(2)}}{\tau_{iz}^{(1)}} = T_i^{(2)}\left(\frac{\tau_i}{\tau_{iz}^{(1)}}, \frac{\tau_j}{\tau_{iz}^{(1)}}, \frac{s_{ij}}{\tau_{iz}^{(1)}}\right) \tag{3.12}$$

Thus, we have expressed the delay and output transition times as functions of temporal parameters only. So far we have not distinguished between the two inputs in any way. However, a key assumption while deriving (3.9) and (3.10) was that the effect of proximity of input transitions should be a perturbation on the delay due to a single "dominant" input. To satisfy this assumption the correct identification of the dominant input among all the inputs is critical. Figure 3-2 explains the way we do this for a two-input NAND gate, similar arguments hold for the NOR gate. The inputs and outputs are shown as piecewise-linear. Consider the case when the slower input arrives first (shown in solid) and the faster input (shown dashed) arrives a little later. The rising waveforms z_a and z_b show the corresponding outputs when each of the inputs is switching by itself. The waveform z_{ab} is the output response due to both inputs. Clearly, it is more appropriate to view input *b* as the dominant one because the time when z_{ab} crosses the V_{il} threshold is closer to the time when z_b crosses V_{il} rather than to the time when z_a crosses V_{il}. This agrees with our notion of proximity being a perturbation of the output produced when *b* alone is switching. Thus, even though *a* crosses the V_{ih} threshold first, it is input *b* that is identified as the dominant one. However, there is a minimum separation equal to $\Delta_{az}^{(1)} - \Delta_{bz}^{(1)}$ after which *a* becomes the dominant input. This is so because beyond this separation, the time when z_a crosses the V_{il} threshold will be closer to the time when z_{ab} crosses the V_{il} threshold. This implies that the minimum separa-

tion could be negative when . Note that this also takes the position of the inputs in the series transistor stack into account since the delays could be different, even for the same transition times for the inputs. Thus, for a given separation between the two inputs and their transition times, we first determine the dominant input and then use (3.11) and (3.12) to determine the delay and output transition times, with respect to the dominant input. Thus, if the original inputs are ordered in terms of which one crosses the V_{ih} threshold first, we find a new ordering in terms of which has the most effect on the output waveform. An analogous argument can be made for the case when the two inputs are rising. Based on Figure 3-2 we can also determine the maximum separation between the two inputs for proximity effects to be considered important. We see that for $s_{ab} > \Delta_{az}^{(1)}$, the transitions on *b* can be ignored and the delay will be the same as when *a* was alone. We define this as the proximity window for *b* to have any effect on the delay. However, *b* may still influence the transition time on *z*. Its is only when $s_{ab} > \Delta_{az}^{(1)} + \tau_{az}^{(1}$, that the effect of *b* can be ignored. This then defines the proximity window for *b* to have any influence on the output transition time. Similar arguments apply when *b* is the dominant input.

Figure 3-3 shows the data obtained from a circuit simulation of the circuit shown in Figure 1-1, with input *c* tied to V_{dd}. The fall time of *a* was fixed at 500ps and the fall time of *b* was fixed at 100ps, 500ps and 1000ps. In each case, s_{ab} was varied from $-(\Delta_{bz}^{(1)} + \tau_{bz}^{(1)})$ to $\Delta_{az}^{(1)} + \tau_{az}^{(1)}$. Also shown is the actual crossover point when the causing input changes, for the case when fall time on *b* is 1000ps. We note that there is a discontinuity in the delay value when the dominant input changes. This is because our reference for measuring delay also changes.

Thus, for a two-input gate our delay and output transition time macromodels are (3.11) and (3.12) where *i* refers to the dominant input. That these two macromodels are indeed functions is apparent from the graphs in Figure 1-2 and Figure 3-3. In fact, based on our preliminary model building efforts for the two-input gate, we can say that closed form analytical forms for these macromodels do exist.

In the next section we describe the modeling approach for gates with more than two inputs.

4 Multi-input temporal proximity model

For an *n*-input gate, equations (3.11) and (3.12) extend in a straightforward way to:

$$\frac{\Delta_{iz}^{(n)}}{\Delta_{iz}^{(1)}} = D_i^{(n)}\left(\frac{\tau_1}{\Delta_{iz}^{(1)}}, \ldots, \frac{\tau_n}{\Delta_{iz}^{(1)}}, \frac{s_{i1}}{\Delta_{iz}^{(1)}}, \ldots, \frac{s_{in}}{\Delta_{iz}^{(1)}}\right) \tag{4.1}$$

$$\frac{\tau_{iz}^{(n)}}{\tau_{iz}^{(1)}} = T_i^{(n)}\left(\frac{\tau_1}{\tau_{iz}^{(1)}}, \ldots, \frac{\tau_n}{\tau_{iz}^{(1)}}, \frac{s_{i1}}{\tau_{iz}^{(1)}}, \ldots \frac{s_{in}}{\tau_{iz}^{(1)}}\right) \tag{4.2}$$

where *i* is the most dominant input. Each of these equations has *2n*-1 parameters. Developing a macromodel involving *2n*-1 parameters can be very hard. A closed analytical form may be impossible to obtain which would force one to use a table-lookup approach. However, the size of these tables involving *2n*-1 dimensions would make them impractical. We need to reduce the number of arguments to these functions in order to make the macromodel construction practical. Since all the quantities in these equations have the unit of time, dimensional analysis fails to reduce the number of arguments. Therefore, we need a way of decomposing these functions in terms of simpler, more manageable functions. A technique for doing this is presented in this section.

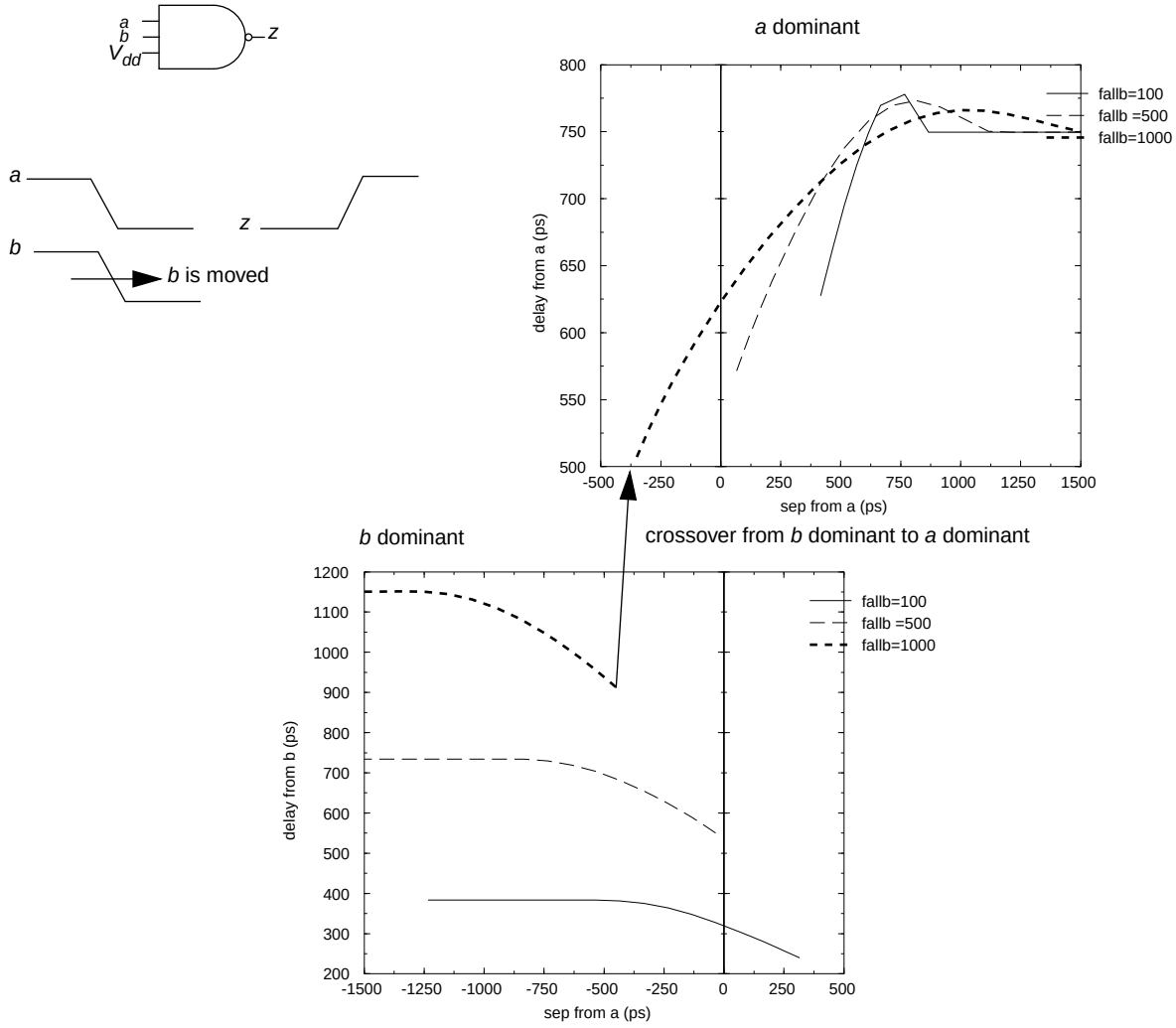


Figure 3-3: Proximity effect on delay

Algorithm *ProximityDelay*:

1. Relabel the inputs to $y_1 \ldots y_n$ such that for any two inputs y_i and y_j, $i < j$ if and only if $s_{y_i y_j} > \Delta^{(1)}_{y_i z} - \Delta^{(1)}_{y_j z}$

2. $i = 2$;

3. **while** ($i \le n$ && $s_{y_1 y_i} < \Delta^{(i-1)}_{y_1 z}$){

4. $\Delta^{(i)}_{y_1 z} = \Delta^{(i-1)}_{y_1 z} + \Delta^{(1)}_{y_1 z}\left[D^{(2)}_{y_1}\left(\frac{\tau_{y_1}}{\Delta^{(1)}_{y_1 z}}, \frac{\tau_{y_i}}{\Delta^{(1)}_{y_1 z}}, \frac{s_{y_1 y_i} + \Delta^{(1)}_{y_1 z} - \Delta^{(i-1)}_{y_1 z}}{\Delta^{(1)}_{y_1 z}} \right) -$

5. $i = i + 1$; } /* end of while */

6. $i = i - 1$;

Figure 4-1: The algorithm for computing delay of a multi-input gate

Our technique for computing the delay and output transition time is based on processing only two inputs at a time, starting from the two most dominant inputs. The algorithm for computing delay is presented in Figure 4-1. A slight modification of the algorithm allows it to be used for output transition time computation. The inputs are reordered in Step 1, based on their dominance, by a straightforward extension of the dual-input case. The reordered inputs are labeled as $y_1 \ldots y_n$. To apply the dual-input macromodel equation (3.11), the cumulative effect of inputs $y_1 \ldots y_{i-1}$ is represented by an equivalent input waveform $y^*(t)$ such that:

$$y^*(t) = y_1(t + \Delta^{(1)}_{y_1 z} - \Delta^{(i-1)}_{y_1 z}) \tag{4.3}$$

where $\Delta^{(1)}_{y_1 z}$ is the delay due to the most dominant input acting alone and $\Delta^{(i-1)}_{y_1 z}$ is the delay due to $y_1 \ldots y_{i-1}$. Equation (4.3) guarantees that the output waveform caused by $y^*(t)$ crosses the delay measurement threshold at exactly the same time that the waveform due to $y_1 \ldots y_{i-1}$ would. The effect of the next dominant input y_i is now accounted for by applying the dual-input proximity macromodel to y^* and y_i:

$$\Delta^{(2)}_{y^* z} = \Delta^{(1)}_{y_1 z} D^{(2)}_{y_1}\left(\frac{\tau_{y_1}}{\Delta^{(1)}_{y_1 z}}, \frac{\tau_{y_i}}{\Delta^{(1)}_{y_1 z}}, \frac{s^*_{y y_i}}{\Delta^{(1)}_{y_1 z}} \right) \tag{4.4}$$

Since this is a shift in the time axis of y_1, we have $\tau_{y^*} = \tau_{y_1}$ and $\Delta^{(1)}_{y^* z} = \Delta^{(1)}_{y_1 z}$. Note that the reference input for $\Delta^{(i)}_{y^* z}$ is y^*; The delay $\Delta^{(i)}_{y_1 z}$ due to $y_1 \ldots y_i$ is easily obtained by changing the reference to y_1 using (4.3):

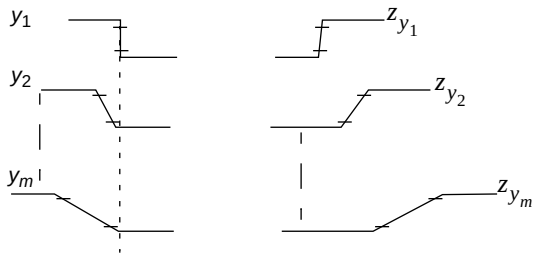


Figure 4-1: An example where our algorithm gives error

$$^{i)}_{1z} = \Delta^{(i-1)}_{y_1 z} + \Delta^{(1)}_{y_1 z}\left[D^{(2)}_{y_1}\left(\frac{\tau_{y_1}}{\Delta^{(1)}_{y_1 z}}, \frac{\tau_{y_i}}{\Delta^{(1)}_{y_1 z}}, \frac{s_{y_1 y_i} + \Delta^{(1)}_{y_1 z} - \Delta^{(i-1)}_{y_1 z}}{\Delta^{(1)}_{y_1 z}} \right) - \right. \tag{4.5}$$

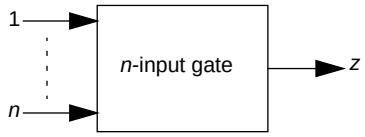
Equation (4.5) clearly shows that the delay due to the *i* inputs that fall in the proximity window is a perturbation of the delay due to the *i*-1 most dominant inputs.

This process is repeated as long as there are inputs within the proximity window, which for the *i*th iteration is given by $\Delta^{(i-1)}_{y_1 z}$. Therefore, if $s_{y_1 y_i} > \Delta^{(i-1)}_{y_1 z}$, we stop processing any more inputs. Implicit in this is the assumption that any input y_j such that *j*>*i* and $s_{y_1 y_j} < \Delta^{(i-1)}_{y_1 z}$, is unimportant. This is reasonable since for this to occur the transition time of y_j must be very slow in which case, it will indeed have very little effect on the output.

It must be emphasized that our algorithm is an approximation of what actually happens. Therefore, while our algorithm computes accurate delays for most cases there are some situations when the incorrect identification of the dominant input leads to significant errors. The primary cause for such errors is the inapplicability of the input ordering based on dominance. There are two cases in which dominance ordering is problematic: 1) when the inputs switch together with identical transition times and 2) when the dominant input arrives very late within the proximity window (see Figure 4-1). In the first case, clearly there is no one input that dominates over others. However, when each input is considered by itself, there will be small differences in delays from each input to the output. Based on this, our algorithm will identify one of the inputs as the dominant one and proceed. This leads to errors, with the maximum error occurring when a step signal is applied to all the inputs at the same time. The only way to accurately model such cases is to take all inputs into account which, as we saw in (4.1) and (4.2), leads to a complicated macromodel.

In the second case, referring to Figure 4-1, we see that the transition times and separations of y_2 through y_m, where y_m is the last input that falls within the proximity window, are such that they affect the output noticeably and y_1 has the effect of merely hastening the output in crossing the delay measurement threshold. In such cases, again, our algorithm underestimates the roles of the other inputs and causes errors.

In order to retain the simplicity of our approach and still get accurate results we added a corrective term to the delay value obtained by our method. We recorded the absolute difference between the delay value computed by our method and the actual delay value, when a step signal is applied to all the inputs at the same time. The correcting factor was bounded from above by this value when $s_{y_1 y_m} \leq 0$, where *m* is the last input that falls within the proximity window. For $s_{y_1 y_m} > 0$, the correcting term was decreased linearly until it became zero for $s_{y_1 y_m} = \Delta^{(m-1)}_{y_1 y_{m-1}}$. A similar correction can be done while computing the output transition time. As we show in the next section this gives satisfactory results.



1)Full model: $$\frac{\Delta_{iz}^{(n)}}{\Delta_{iz}^{(1)}} = D_i^{(n)}\left(\frac{\tau_1}{\Delta_{iz}^{(1)}}, \ldots, \frac{\tau_n}{\Delta_{iz}^{(1)}}, \frac{s_{i1}}{\Delta_{iz}^{(1)}}, \ldots, \frac{s_{in}}{\Delta_{iz}^{(1)}}\right)$$

2) Compositional model:

(a)

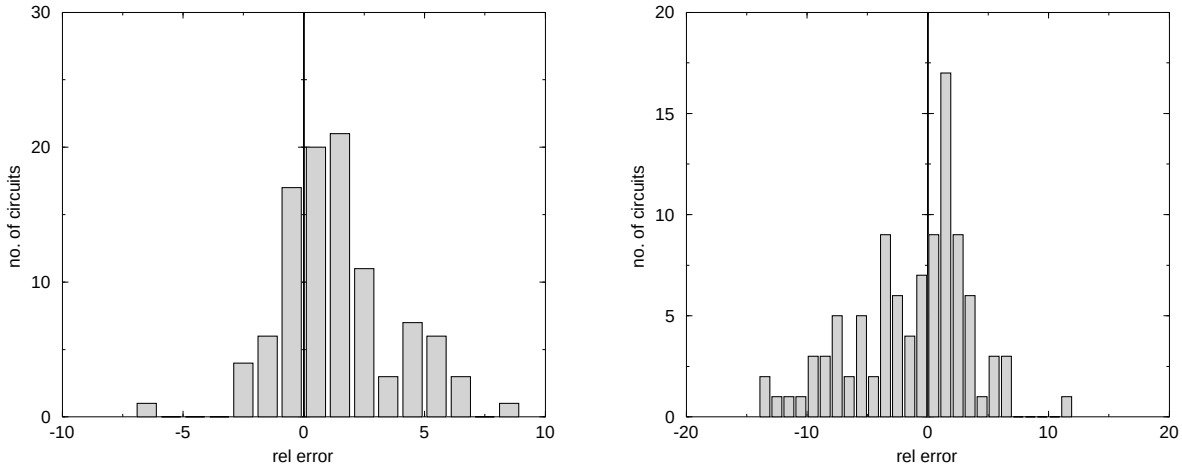
$$\begin{bmatrix} D_{1z}^{(1)} & D_{12z}^{(2)} & \ldots & D_{1nz}^{(2)} \\ D_{21z}^{(2)} & D_{2z}^{(1)} & \ldots & D_{2nz}^{(2)} \\ \ldots & \ldots & \ldots & \ldots \\ D_{n1z}^{(2)} & \ldots & \ldots & D_{nz}^{(1)} \end{bmatrix}$$

(b)

$$\begin{bmatrix} D_{1z}^{(1)} & D_{1z}^{(2)} & \ldots & D_{1z}^{(2)} \\ D_{2z}^{(2)} & D_{2z}^{(1)} & \ldots & D_{2z}^{(2)} \\ \ldots & \ldots & \ldots & \ldots \\ D_{nz}^{(2)} & \ldots & \ldots & D_{nz}^{(1)} \end{bmatrix}$$

Figure 4-2: The storage complexity of our approach

The computational complexity of our method is dominated by the memory requirements since it is clear that the computation time is not significant here. We consider the storage requirements for computing delay. Exactly similar results hold for the output transition time. Consider the *n* input gate shown in Figure 4-2. The various modeling options are also shown in the figure. If a full model of the form (4.1) is used, we will require *n* functions of 2*n*-1 arguments for delay. However, we have already noted the difficulties of such a model and we consider the compositional model introduced in this paper next. Although, so far we have used $D_{iz}^{(2)}$ to denote the dual-input macromodel, in practice, this actually represents a family of functions, one for each input pair. This is shown in the form of a matrix in 2(a) in Figure 4-2. Here, $D_{ijz}^{(2)}$ denotes the dual input macromodel of the form (3.11) with $i \neq j$ and $D_{iz}^{(1)}$ denotes the single-input macromodel of the form (3.7). The arguments of the functions have been omitted for clarity. From the matrix in 2(a), it is clear that we need *n* single input macromodels and $n^2 - n$ dual-input macromodels. However, our efforts in constructing the dual-input macromodels show that we need only *n* such macromodels, one for each input being the dominant one. This is shown in 2(b) of the figure. Thus, we require at most *n* macromodels for the single-input case and *n* macromodels for the dual-input case, making it 2*n* macromodels to handle proximity effect on delay. Additional 2*n* macromodels are required for the output transition time.



(a) Relative error distribution for delay

(b) Relative error distribution for rise time

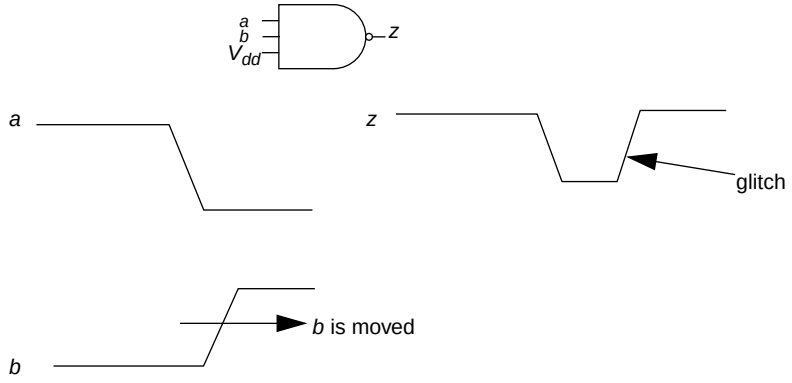
Figure 5-1: Error distribution for delay and output rise time

5 Experimental validation

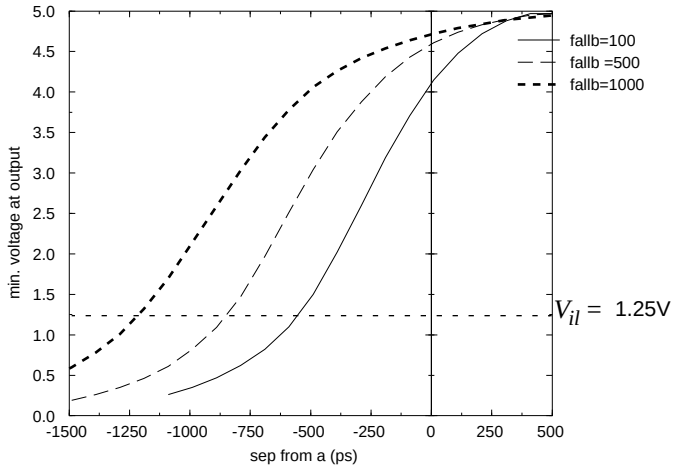
In order to validate our approach, we simulated the circuit in Figure 1-1 for a range of input separations and transition times. The fall times of the three inputs were varied from 50ps to 2000ps. The separation between *a* and *b* and between *a* and *c* were varied from -500ps to 500ps. Note that this automatically varies the separation between *b* and *c* as well. The window size was chosen to ensure that all three inputs are influential in determining the output. In order to precisely control the separations and rise times of the inputs, piecewise-linear inputs were used.The transistor sizes and the load capacitance were fixed at the values shown. We used HSPICE as the macromodel for processing the dual-input case. A total of 100 different input configurations were randomly generated and simulated. We compared the delay and rise time computed by our algorithm with values obtained through simulation. The results are summarized in Table 5-1and the corresponding bar charts showing the error distribution are given inFigure 5-1. We observe that in most cases the delay computed by our technique was within $\pm 5\%$ and the

Table 5-1: Comparison of model with circuit simulation

Quantity	Delay	Rise time
Mean error	1.4%	-1.33%
Std-dev	2.46%	4.82%
Max error	8.54%	11.51%
Min error	-6.94%	-13.15%



(a) Proximity effect due to opposite going transitions



(b) Magnitude of output glitch versus separation

Figure 6-1: Relationship between inertial delay and proximity effect

output rise time was within $\pm 10\%$. Note that the larger error in output transition times can be tolerated since the effect of output transition time gets attenuated by the gain of the following stage [10].

6 Inertial delay and proximity effect

We now digress briefly to show the relationship between inertial delay and proximity effect. Referring to Figure 6-1(a), assume input *a* falls and input *b* rises. This will generate a negative going glitch at the output if the two inputs switch in close temporal proximity to each other. Figure 6-1(b) shows the magnitude of the minimum output voltage as a function of separation between *a* and *b*, obtained by simulating the circuit in Figure 1-1, with input *c* tied to V_{dd}. The fall time for input *a* was fixed at 500ps and the rise time on *b* was fixed at 100ps, 500ps and 1000ps. The dotted horizontal line shows the V_{il} threshold for the

circuit. Only when the magnitude of the output voltage falls below this value, can we conclude that the output has completed a transition. We see that when input *b* comes much earlier than input *a*, the output completes its falling transition. However, when *a* and *b* switch close together, the falling transition on *a* blocks the effect caused by the rising transition on *b*. Thus, there must be a minimum separation between *a* and *b* for the output to complete its transition. We can model this as follows. We first find a macromodel for the minimum voltage at the output which will be similar to (3.9). Here *i* would refer to the non-controlling input (*b* in this example). From this equation, we find the minimum separation at which the magnitude of voltage is equal to V_{il}. This is the minimum separation between two inputs of opposite transitions that will generate a valid output. Note that for a NAND gate, we can have a rising glitch at the output only when the same input first falls and then rises. We can have a separate macromodel for the maximum voltage in this case.

7 Conclusions

We have shown that the temporal parameters of the inputs such as their transition times and their arrival times with respect to each other have a significant effect on the delay of a multi-input gate. Since, for a multi-input gate, we have a family of VTCs to choose the thresholds from, it raised the question of choosing appropriate thresholds to measure delay. It was shown that choosing the minimum V_{il} and the maximum V_{ih} from among the thresholds obtained by all possible VTCs of the gate, ensured that the delay would never become negative for any input situation. We next showed how the non-temporal factors affecting the delay can be captured by one parameter and this was shown to simplify the form of the delay and output transition time macromodel equations for a two-input gate. Our conjecture that proximity delay is a perturbation of the delay due to a single input transition enabled us to derive these macromodels as functions of three arguments. This created the notion of a dominant input, identification of which led to the development of dual-input proximity model. Following this, we presented a novel technique for calculating the delay of a multi-input gate by repeated application of the dual-input proximity model. Simulation results show that this technique works quite well in practice. The results are more accurate than previously published methods of calculating delay for multi-input gates which rely on the reduction of the gate to an equivalent inverter. An added advantage of our method is that it is not limited to CMOS technology alone. Finally, we showed how the inertial delay of a gate arises as a consequence of the proximity effect and an approach for capturing the inertial delay of a two-input gate was suggested.

Our future efforts will seek to provide a comprehensive delay model for multi-input gates. This will include single and dual-input macromodels for delay and output transition times with respect to each input. We also plan to use this technique for the CGaAs [1] technology.

References

[1] J. K. Abrokvah et al, “A Manufacturable Complementary GaAs Process”, *IEEE GaAs IC Symposium*, pp:127-130, 1993.

[2] J. R. Burns, “Switching Response of Complimentary-Symmetry MOS Transistor Logic Circuits”, *RCA Review*, 25(12):627-661.

[3] H.-Y. Chen and S. Dutta, “A Timing Model for Static CMOS Gates,” in *IEEE Conference on Computer-Aided Design*, 1989.

[4] S. Dutta, S. S. M. Shetty and S. L. Lusky, “A Comprehensive Delay Model for CMOS Inverters”, *IEEE Journal of Solid-State Circuits*, 30(8):864-871, 1995.

[5] C. T. Gray, W. Liu, and R. K. Cavin III, “Exact Timing Analysis Considering Data Dependent Delays,” Technical Report NCSU-VLSI-92-04, North Carolina State University, December 1992.

[6] N. Hedenstierna and K. O. Jeppson, “CMOS Circuit Speed and Buffer Optimization”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 6(2):270-281, 1987.

[7] D. A. Hodges and H. J. Jackson, “Analysis and Design of Digital Integrated Circuits”, McGraw-Hill, 1983.

[8] Y.-H. Jun, K. Jun, and S.-B. Park, “An Accurate and Efficient Delay time Modeling for MOS Logic Circuits using Polynomial Approximation,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9(6):1027-1032, 1989.

[9] A. I. Kayssi, “Timing Macromodels for Digital Circuits,” Ph.D., University of Michigan, EECS Dept., 1993.

[10] A. I. Kayssi, K. A. Sakallah and T. Mudge, “The Impact of Signal Transition Time on Path Delay Computation”, *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, 40(5):302-309, 1993.

[11] J. T. Kong and D. Overhauser, “Methods to Improve Digital MOS Macromodel Accuracy”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(7):868-881, 1995.

[12] Meta-Software, “Hspice H92A User's Manual,”,

[13] A. Nabavi-Lishi and N. C. Rumin, “Inverter Models of CMOS Gates for Supply Current and Delay Evaluation,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(10):584-594, 1994.

[14] T. Sakurai and A. R. Newton, “Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas,” *IEEE Journal of Solid-State Circuits*, 25(4):584-594, 1990.

[15] T. Sakurai and A. R. Newton, “Delay Analysis of Series Connected MOSFETs”, *IEEE Journal of Solid-State Circuits*, 26(2):122-131,1991.

[16] E. Seewann, “Switching Speeds of MOS Inverters”, *IEEE Journal of Solid-State Circuits*, 15(4):246-252, 1980.

[17] S.-Z. Sun, D. H. C. Du, and H.-C. Chen, “Efficient Timing Analysis for CMOS Circuits Considering Data Dependent Delays,” in *Proc. IEEE International Conference on Computer Design (ICCD)*, Cambridge, Massachusetts, 1994.

[18] TACTIC User and Reference Manual, Cascade Design Automation Corporation, 1994.