The Impact of Signal Transition Time on Path Delay Computation

Ayman I. Kayssi, *Member, IEEE,* Karem A. Sakallah, *Senior Member, IEEE,* and Trevor N. Mudge, *Senior Member, IEEE*

***Abstract*—It has been recognized for some time that nonzero signal rise and fall times contribute to gate propagation delays. Practically, however, most timing analysis tools ignore these contributions when computing path delays and identifying critical paths in combinational circuits. In this paper we describe how these rise and fall times can be incorporated into path analysis algorithms. Interestingly, we show that signal transition time information can be accounted for in a simple and efficient preprocessing step followed by the application of standard path analysis methods. This two-step approach is shown to predict path delays with sufficient accuracy without unnecessarily complicating path analysis.**

I. Introduction

TIMING analysis is now recognized as an important tool for verifying the temporal correctness of digital circuits [8]. By ignoring the logical function of the gates and representing them exclusively by their delay properties, timing analysis allows us to compute path delays with an accuracy approaching that of detailed circuit simulation at a speed exceeding that of logic simulation. This speed advantage, however, comes with the disadvantage of identifying logically impossible paths as the performance-limiting paths in a circuit. This so called *false path* problem is the major source of inaccuracy in timing analysis; it has received, and continues to receive, a great deal of attention and has yet to be resolved satisfactorily [16], [12], [6], [4]. Even when false paths have been correctly identified and eliminated, timing analysis can still produce inaccurate path delays if its gate delay models are inadequate. In this paper we focus on this second source of inaccuracy. Specifically, we show that nonzero signal transition times have a noticeable impact on the accuracy of timing analysis and show how they can be incorporated in gate and path delay computation.

For combinational circuits, timing analysis tools compute path delays and help in the identification of short and long critical paths. The analysis is usually carried out in two distinct phases:

- A *delay modeling* phase in which the delays of individual gates and wires are calculated. The delay models are typically derived analytically or constructed from extensive measurements or circuit simulations.
- A *path delay* calculation phase in which the individual *fixed* gate and wire delays are combined to form path delays. The most common approaches for path delay calculation are the *critical path method* (CPM) [15] and its variants.

The development begins in Section II with a careful classification of gate delay models—based on whether they include transition time effects—as either static or dynamic. We then define gate delay and output transition time functions and explore their properties. In Section III, two distinct approaches for including the effects of input transition time in path delay analysis are presented. The first approach, called *extended CPM*, extends the standard fixed-delay CPM technique to the case of delay functions of signal transition time. The second approach, dubbed *context delay* modeling, accounts for signal transition time effects in a pre-processing step that produces *fixed* delays which are fed to a standard CPM tool. In Section IV, we present an experimental comparison of these approaches on several benchmark circuits and the paper ends with some conclusions in Section V.

II. Dynamic Gate Delay Models

Key to the inclusion of transition time effects in path delay computation is a careful definition of gate delay. This section establishes a classification of gate delay models and identifies the functional relationship between gate delay and input transition time.

Timing analysis and logic simulation are based on the premise that signal delay through logic gates can be separated out and "lumped" outside the gates. Of course, such lumped delay is merely a convenient abstraction of the underlying electrical behavior which is described by a system of nonlinear ordinary differential equations. Fortunately, the abstraction happens to be reasonably valid for a wide range of digital-mode operation and is universally used.

Without loss of generality, we will investigate the delay models of multiple-input/single-output logic gates. The lumped delay abstraction characterizes the input/output gate behavior as a composition of two types of functions:

1) *Instantaneous combining functions* C, which operate on all gate inputs to produce the output. For logic simulation, the combining functions are Boolean switching

Manuscript received May 21, 1992; revised January 5, 1993. This work was supported in part by the Defense Advanced Research Projects Agency under Grant DAAL03-90-C-0028 and by the National Science Foundation under Grant MIP-9014058. This paper was recommended by Associate Editor G. De Micheli.

A.I. Kayssi is with the Department of Electrical Engineering, American University of Beirut, New York, NY 10022.

K.A. Sakallah and T.N. Mudge are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109.

IEEE Log Number 9207956.

1057–7130/93$03.00 © 1993 IEEE

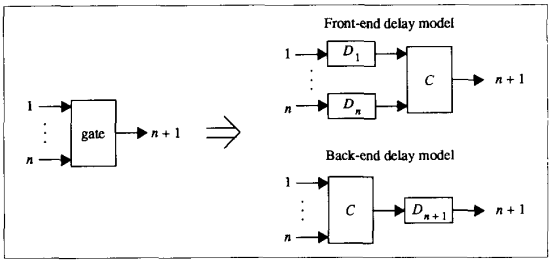


Fig. 1. Front- and back-end delay models.

(0-1) or multivalued logic functions. For timing analysis, they are min and max functions of the input signal arrival times.

2) *Delay functions* D_i, which translate signal transitions forward in time.

Fig. 1 shows the two possible configurations for these functions. In the *front-end delay* model, the delay functions precede the combining function; they are applied to the gate inputs whose delayed versions are then combined to produce the gate output. In the *back-end delay* model, the delay function follows the combining function which is applied directly to the gate inputs. Because it uses more delay functions, front-end delay gives greater modeling flexibility than back-end delay. Most logic simulators and timing verifiers use back-end delay, however, because of its lower storage requirements.

We consider next the possible choices for the delay functions. We restrict the discussion to deterministic models of pure propagation (transport) delay [2]; statistical and inertial delay models are beyond the scope of the current investigation. For purposes of classifying different delay assumptions, it is useful to identify the important factors which affect gate delay. These can be grouped as follows.

1) *Circuit* parameters, such as transistor sizes, capacitive loads, and fanouts.
2) *Process* parameters, such as oxide thickness and threshold voltages.
3) *Environmental* parameters, such as temperature and supply voltage.
4) *Wave shape* of the input switching signal, usually captured by its nonzero transition time as measured between appropriate voltage thresholds.
5) *Temporal proximity* of the transitions on different inputs, i.e., the degree of overlap among these transitions.

For timing analysis purposes, the parameters in the first three groups are usually invariant during normal circuit operation; those in the last two are not. Thus, if nonzero transition time and input proximity effects are neglected, the delay model reduces to a constant function. Because of its simplicity this is the most commonly used model. However, this simplification can lead to significant errors in the gate delay [14]. The effects of varying input transition time are illustrated in Fig. 2. The response of a CMOS inverter and a GaAs DCFL (Gallium Arsenide Direct-Coupled FET Logic) [10] inverter to fast and slow rising signals was obtained using the HSPICE [13] circuit simulator. In both cases a doubling of the input rise time is

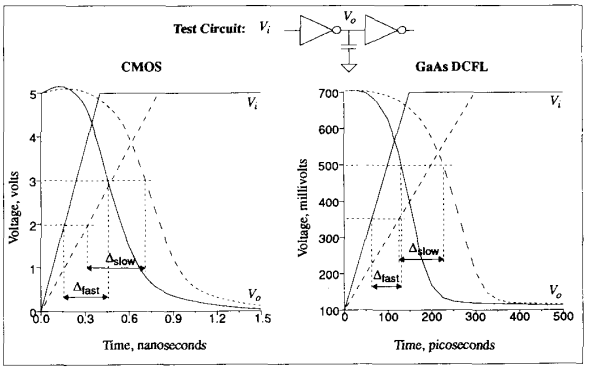


Fig. 2. Effect of input transition time on gate delay.

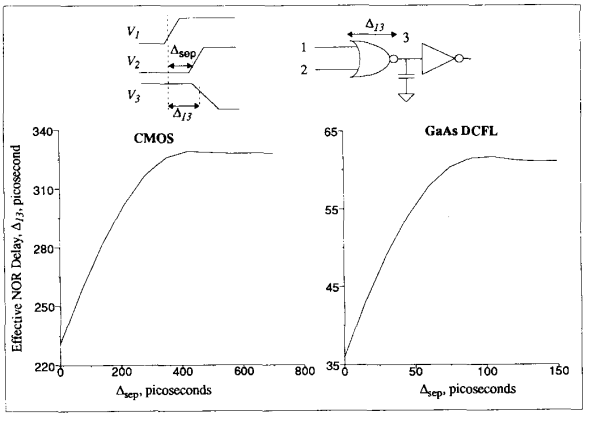


Fig. 3. Effect of temporal proximity on gate delay.

seen to cause an increase in the inverter delay: 35% for CMOS and 45% for GaAs. Fig. 3 depicts the change in gate delay due to the proximity of a transition on a second input. For both a CMOS NOR gate and a GaAs DCFL NOR gate, we observe that as the separation between the transitions on the two inputs is decreased from about one gate delay to 0, the *effective* delay from the earlier input decreases by 30% for CMOS and 41% for GaAs.

A. Model Classification

These observations suggest the following classification of gate delay models based on the effects they account for:

1) *Static* delay models which account for the invariant parameters (such as load capacitance) but ignore both signal transition time and input proximity effects. These models lead to *constant* delay functions.
2) *Dynamic* delay models which not only account for all of the invariant parameters, but also for signal transition time and input proximity effects. These models can be classified further into:
   a. Models which account for signal transition time but ignore input proximity effects. Thus, they are based on a single-input-change assumption.
   b. Models which account for both signal transition time and input proximity effects.

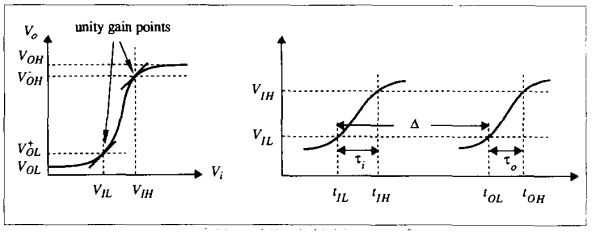


Fig. 4. Definitions of threshold voltages and gate delay.

Clearly, when dynamic effects are important, a delay model which takes them into account will yield more accurate gate delays than a static delay model. In particular, for circuits which have global busses as well as short interconnection runs, it is not uncommon for signal transition times to have a 10-to-1 spread. In such cases, a static delay model can seriously underestimate gate delay.

In this paper we restrict the discussion to gate delay functions which account for nonzero signal transition times and show how they can be incorporated in path delay analysis. A thorough investigation of proximity effects requires further research.

B. Definition of Delay and Transition Time Functions

It is appropriate at this point to clearly define what is meant by gate delay. Fig. 4 shows the dc transfer curve and typical input and output waveforms for a noninverting buffer. The input thresholds at which the differential gain of the buffer is unity are referred to as V_{IL} and V_{IH} [7]. These two thresholds serve to define the reference times on the input and output voltage waveforms for measuring delay: t_{IL} and t_{OL} are the input time and corresponding output time at which V_i and V_o cross V_{IL}, t_{IH} and t_{OH} are the input time and corresponding output time at which V_i and V_o cross V_{IH}. Rising gate *propagation delay* Δ is now defined as the time interval between t_{IL} and t_{OL}. The *transition time* of V_i, τ_i, is defined as the time interval between t_{IL} and t_{IH}; the *transition time* of V_o, τ_o, is defined as the time interval betwen t_{OL} and t_{OH}. Our choice of the above thresholds for measuring propagation delay and transition times, unlike other more commonly used thresholds such as the 50% level of the signal swing for delay and the 10% and 90% levels for transition time, insures that Δ will always be positive. This fact is easily established by observing that V_o, whose initial value is V_{OL}, can never cross the V_{IL} threshold *before* V_i does since the differential gain of the gate for $V_i < V_{IL}$ is less than 1 and $V_{OL}^{+} < V_{IL}$. A similar argument holds for falling outputs.

We can now describe a single-input-change dynamic delay model by two functions which depend on the input transition time τ_i: a propagation delay function D, and an output transition time function T. Symbolically,

$$\begin{aligned} \Delta &\equiv t_{OL} - t_{IL} = D(\tau_i) \\ \tau_o &\equiv t_{OH} - t_{OL} = T(\tau_i). \end{aligned} \tag{1}$$

These definitions pertain to rising signals at the input and output of a noninverting gate. They are easily extended to the case of falling signals and inverting gates. In addition, if the rising and falling delays are significantly different they should be modeled by separate functions.

C. Properties of Delay and Transition Time Functions

Before showing how this delay model is used in path delay computation, we examine now some of the properties of the D and T functions. The results in this section apply for both the D and T functions, so any discussion of the D function is also valid for the T function. We seek to understand the dependence of Δ and τ_o on τ_i, which, in general, is technology-specific. However, there are certain aspects of the functions that are common to all technologies and only depend on the "shape" of the dc transfer curve, namely that it has both zero-gain and high-gain regions. This is easily revealed by a first-order analysis which ignores all circuit capacitances, and assumes piece-wise linear forms for the transfer curve and signal waveforms at the input and output of a noninverting buffer. Thus, during the transition interval, the output voltage V_o is related to the input voltage V_i by

$$V_o = V_{OL} + K(V_i - V_{IL}) \tag{2}$$

where $K \equiv (V_{OH} - V_{OL})/(V_{IH} - V_{IL})$ is the stage gain. Applying the input waveform

$$V_i = V_{IL} + \frac{V_{IH} - V_{IL}}{\tau_i} t \tag{3}$$

we can now obtain the following output waveform:

$$V_o = V_{OL} + K(V_{IH} - V_{IL})\frac{t}{\tau_i} \tag{4}$$

which allows us to express the buffer delay Δ and output transition time τ_o as

$$\frac{\Delta}{\tau_i} = \left(\frac{V_{IL} - V_{OL}}{V_{IH} - V_{IL}}\right)\frac{1}{K} \tag{5}$$

and

$$\frac{\tau_o}{\tau_i} = \frac{1}{K}. \tag{6}$$

This technology-independent simplified analysis shows that the delay and output transition time increase linearly with the input transition time. In the presence of capacitances, we can derive two properties for the delay and transition time functions, and show the above result to be a special case of the general results.

The first property of the D and T functions is that they increase monotonically with τ_i. This is clearly shown in Fig. 5 where the propagation delays of CMOS and GaAs DCFL inverters are plotted against the input transition time for three different values of load capacitance (0 fF, 100 fF, and 300 fF.) For GaAs DCFL, both rising and falling delays were measured, while for CMOS, which has similar rise and fall characteristics, only the delay of the falling output waveform was measured. The delay measurements were taken from the output of HSPICE circuit simulations. In all cases, the delay

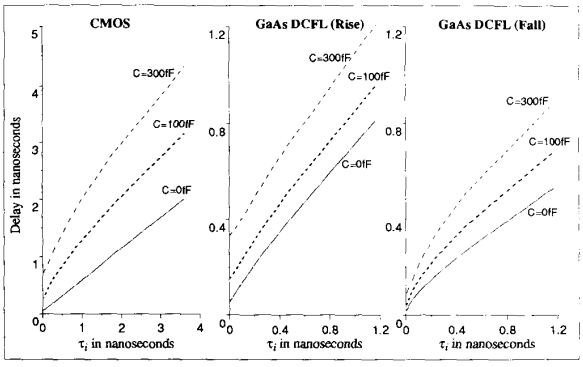


Fig. 5. Gate delay variation with input transition time.

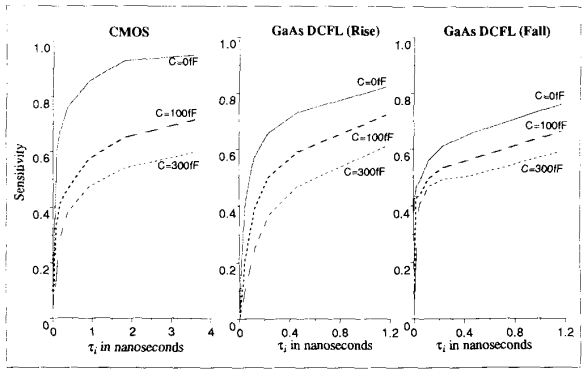


Fig. 6. Sensitivity of gate delay with respect to τ_i.

increases monotonically with the input transition time, and is asymptotically linear for large values of τ_i. The slopes of the linear portions of the delay curves all approach the same value, which is approximately equal to the value given by (5). For small values of τ_i, the delay function is slightly nonlinear, with a positive y-axis intercept. The behavior of the delay function for small values of τ_i depends on the specific circuit technology, and varies according to the topology of the output stage of the gate (complementary, depletion pull-up, resistive pull-up, etc.) [11].

The second, and more important property of delay and output transition time functions is that their sensitivity to changes in the input transition time is always nonnegative and less than 1. Sensitivity of a function $f(x_1, x_2, \cdots, x_n)$ to the variable x_i is defined as [5]

$$S_f^{x_i} = \frac{x_i}{f(x_1, x_2, \cdots, x_n)} \frac{\partial f}{\partial x_i}. \tag{7}$$

Fig. 6 shows the sensitivity of the gate delay functions in Fig. 5 to input transition time; in all cases, the sensitivity is seen to lie in the interval [0, 1). A nonnegative, less than unity sensitivity means that if the input transition time were to increase by $\rho\%$ ($\rho > 0$), the propagation delay and output transition times would both change by $\phi\%$ and $\gamma\%$, respectively, with ϕ and γ satisfying the inequalities

$$\begin{aligned} 0 < \phi < \rho \\ 0 < \gamma < \rho. \end{aligned} \tag{8}$$

Analytically, we can prove that the sensitivity is always nonnegative and less than 1. As τ_i approaches 0, the sensitivity goes to 0, independent of the value of load capacitance. This is due to the definition of sensitivity (7), and to the fact that for zero input transition time, the delay is nonzero, and its derivative is finite. For large values of τ_i, we can assume a linear equation for the delay Δ as a function of input transition time τ_i:

$$\Delta = a_0 + a_1 \tau_i. \tag{9}$$

The constant a_0 depends on the value of load capacitance, as is clear from Fig. 5, and the value of the constant a_1 is given by (5). In this range, the sensitivity is given by

$$S_\Delta^{\tau_i} = \frac{a_1 \tau_i}{a_0 + a_1 \tau_i} = \frac{1}{1 + \frac{a_0}{a_1 \tau_i}}. \tag{10}$$

Therefore, as $\tau_i \to \infty$, $S_\Delta^{\tau_i} \to 1$. We note that the delay is less sensitive to variations in input transition time when the ratio a_0/a_1 is large. This is the case when the capacitive load and/or the gain K of the stage is large.

To conclude, it was shown that the effect of input transition time on gate delay and output transition time always gets attenuated regardless of the magnitude of τ_i. For very small values of τ_i, the D and T functions are not sensitive to variations in input transition time. For large values of τ_i, the sensitivity approaches 1. The range for which the sensitivity is high depends on the capacitive loading and the dc gain of the gate.

III. Path Delay Analysis Using Dynamic Gate Models

In this section we propose two approaches for incorporating nonzero transition time effects in path delay analysis. In the first approach, path analysis algorithms are extended to handle nonzero signal transition times, and to propagate their effects in the circuit using the dynamic gate delay and transition time functions. The second approach, dubbed context delay modeling, accounts for nonzero transition time effects in a preprocessing step which computes fixed gate delays that are subsequently fed to standard path analysis tools. This alternative approach is slightly less accurate; it is attractive, however, when it is undesirable or impossible to modify path analysis tools. Experimental comparisons of the two approaches are given in Section IV. Throughout, we confine our discussion to long path delay analysis. Short path analysis is a straightforward adaptation which, for the most part, is limited to replacing max functions with min functions. In the discussion that follows we will note those places where short path analysis is more involved.

A. Extending Path Analysis Algorithms

The primitive operation in CPM-based algorithms for long path delay computation is a max function on signal arrival times. Thus, we must first associate appropriate arrival times

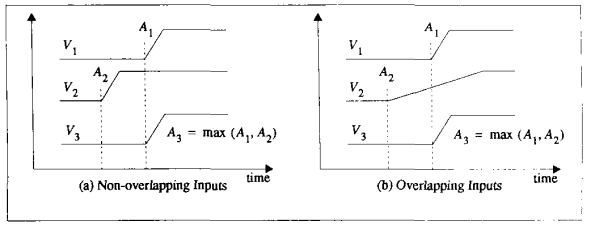


Fig. 7. Max and proximity.

to signals with nonzero transition times. To be consistent with our earlier definition of gate delay, we define the arrival time of a rising/falling signal as the time at which it crosses V_{IL}/V_{IH}. The max operation then chooses the input signal which crosses its threshold last; this is depicted in Fig. 7. Note, however, that this operation is meaningful only if the input signals are nonoverlapping. When the input transitions overlap [Fig. 7(b)], the max operator selects V_1 as the later signal even though the transition of the "earlier" signal V_2 outlasts it due to a much longer transition time. In fact the behavior will be more complex as the results of Fig. 3 indicate. A more accurate model of the max operation would account for input proximity effects, but is beyond the scope of this paper.

Consider now an n-input gate which is characterized by n propagation delay functions $D_i(\tau_i)$ and n transition time functions $T_i(\tau_i)$. Furthermore, let $W_i(\tau_i)$ denote the *minimum* temporal separation required between a transition on input i and a subsequent transition on any other input for proximity effects to be nonexistent; based on the results in Fig. 3, we assume that $W_i(\tau_i) \geq D_i(\tau_i)$. Denoting the arrival and transition times at the gate output by A_o and τ_o, we can now summarize the computation performed in the CPM front-end and back-end delay models as follows.

- Front-end Delay Model:

$$A_o^F = \max_i (A_i + D_i(\tau_i)) \equiv A_k + D_k(\tau_k) \tag{11}$$

$$\tau_o = T_k(\tau_k) \tag{12}$$

where k is some value of $1 \leq i \leq n$.

- Back-end Delay model:

$$A_o^B = \max_i (A_i) + D_m(\tau_l) \tag{13}$$

$$\tau_o = T_o(\tau_l) \tag{14}$$

where l is the index of the "late" input and

$$D_m(\tau) \equiv \max_i [D_i(\tau)] \tag{15}$$

$$T_o(\tau) = T_m(\tau) \tag{16}$$

and m is the index of the input with the "longest" delay function, i.e., whose delay function satisfies $D_m(\tau) \geq D_i(\tau)$, for all $\tau \geq 0$.

For proximity effects to be absent, we require that the input transitions be separated in time by W_i, i.e, the time intervals $\{\langle A_i, A_i + W_i(\tau_i)\rangle\}$, for $i = 1, \cdots, n$, should not overlap. If this *nonoverlap condition* is met, it is easy to see that the front-end arrival time of (11) is bounded from above by the back-end arrival time of (13):

$$A_o^F \leq A_o^B \tag{17}$$

because in the case of nonoverlap, $k = l$.

It should be evident that the front-end delay model can represent the propagation properties of the gate more faithfully than the back-end delay model. For example, if the propagation delays from different inputs to the gate output are different (as is the case for a NAND gate in CMOS), a back-end model must necessarily use the propagation delay function corresponding to the "worst" input line if it is to avoid underestimating the long path delay. Unfortunately, this can cause it to excessively overestimate the delay in some cases, as (17) suggests. A front-end model, on the other hand, uses different delay functions for each input line, and can be made to produce results which are closer to the actual gate behavior. Both models have been implemented in an experimental path analysis tool, DExtr (*D*elay *Extr*actor), and their results on several benchmark circuits are compared in Section IV.

The front- and back-end delay models can be defined for the case of the short path by replacing max with min, in particular redefining D_m to be the "shortest" delay function, i.e., whose delay function satisfies $D_m(\tau) \leq D_i(\tau)$. Similarly, the front-end delay model gives a tighter bound on the short path because it can be shown that the earliest signal arrival time on the gate output in the back-end case is a lower bound on the earliest arrival time on the gate output in the front-end case.

B. Context Delay Modeling

Equation (10) shows that the sensitivity of gate propagation delay to input signal transition time is less than 1. The same is also true for the sensitivity of the output transition time. This means that the effect of τ_i diminishes rather quickly after only a few stages of logic. Thus, while transition time effects are significant, they are also localized. In a chain of n identical gates, the effect of the transition time at the input of the first gate on the delay of the last gate is reduced by a factor of $(1/S)^n$, where $S < 1$. This indeed is confirmed by the data in Fig. 8 which shows that the effect on gate delay of doubling the input transition time diminishes very quickly from 45% to less than 1% after only two levels of logic.

The delay of a specific gate can thus be computed by only considering gates that are its close predecessors. Specifically, accurate delays can be computed by considering the "local context" of the gate. Delay models can thus be classified according to how much context they account for in computing gate delay. A *c-context* model includes c levels of predecessor gates in the computation of gate delay. Thus, static delay models can be viewed as 0-context models; 1-context and 2-context dynamic models include, respectively, one and two levels of predecessor gates.

TABLE I
LONG PATH DELAYS FOR BENCHMARK CIRCUITS

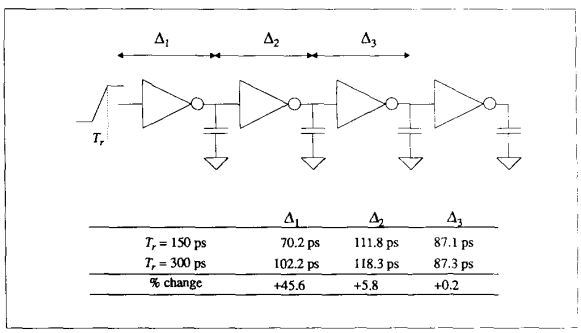
Circuit (#I, #G, #O)[a]	# Gates on Critical Path	τ^b_{avg}	Path Delay, picoseconds (% error relative to HSPICE)						
			0-Context	1-Context		2-Context		Extended CPM	HSPICE
			BE[c]	FE[d]	BE	FE	BE	FE & BE	
Control1 (17, 23, 5)	10	53	1032 (−0.96%)	1033 (−0.86%)	1048 (0.58%)	1044 (0.19%)	1055 (1.25%)	1045 (0.29%)	1042
Control2 (26, 106, 41)	8	101	1325 (−7.15%)	1381 (−3.22%)	1381 (−3.22%)	1414 (−0.91%)	1414 (−0.91%)	1420 (−0.49%)	1427
Control3 (10, 41, 14)	6	87	825 (−8.94%)	873 (−3.64%)	873 (−3.64%)	895 (−1.21%)	895 (−1.21%)	898 (−0.88%)	906
Control4 (6, 8, 2)	6	50	614 (2.50%)	590 (−1.5%)	590 (−1.50%)	591 (−1.34%)	591 (−1.34%)	592 (−1.17%)	599
Control5 (22, 62, 8)	10	91	1443 (−8.90%)	1546 (−2.4%)	1557 (−1.70%)	1587 (0.19%)	1600 (1.01%)	1591 (0.44%)	1584
Comb3 (2, 22, 15)	8	148	1496 (−26.20%)	1934 (−4.59%)	1935 (−4.54%)	2003 (−1.18%)	2003 (−1.18%)	2016 (−0.54%)	2027
Bypass3 (32, 157, 2)	14	80	1960 (−5.68%)	2029 (−2.36%)	2121 (2.07%)	2089 (0.53%)	2212 (6.45%)	2076 (−0.10%)	2078
Bypass4 (32, 184, 2)	14	97	2271 (−5.02%)	2325 (−2.76%)	2359 (−1.34%)	2392 (0.04%)	2461 (2.93%)	2391 (0.0%)	2391

[a] #I: number of inputs; #G: number of gates; #O:number of outputs.

[b] Average transition time on critical path (picoseconds).

[c] BE: back-end model.

[d] FE: front-end model.



	Δ_1	Δ_2	Δ_3
T_r = 150 ps	70.2 ps	111.8 ps	87.1 ps
T_r = 300 ps	102.2 ps	118.3 ps	87.3 ps
% change	+45.6	+5.8	+0.2

Fig. 8. Diminishing effect of signal transition time.

```
Algorithm ContextDelay:

Given: Combinational gate network, N
       Number of context levels, c
       Characteristic (typical) transition time, τc
       Gate delay and transition time functions, Di(τi) and Ti(τi) for each gate input

Initialization: Sort the gates in topological order; let L be the number of logic levels

for l = 1 to (L - c) {                          /* Scan network in topological sort order */
    Set transition time for all inputs to gates at level l to τc;
    for k = l to (l + c - 1) {                  /* Look forward c levels */
        for each gate g at level k {
            for each input i to gate g
                τgi = Ti(τi);                   /* Compute output transition times for each distinct input */
            τg = maxi(τgi);                     /* Save max output transition time */
        }
    }
    for each gate g at level (l + c) {          /* Compute context delay for gates at level (l + c) */
        for each input i to gate g
            Δgi = Di(τi);                       /* Use separate input delays for front-end model */
        Δg = maxi(Δgi);                         /* Use max delay for back-end model */
    }
}
```

Fig. 9. Context modeling algorithm.

Algorithm *ContextDelay* in Fig. 9 sketches the essential steps needed to compute the c-context delays for all gates in a combinational circuit.[1] The basic concept in this algorithm is to sweep a window which is c gate levels wide across the circuit. The transition times at the inputs of this window are set equal to a characteristic time τ_c, and for each major iteration, the c-context delays of the gates at the last level in the window are calculated.

IV. EXPERIMENTAL RESULTS

We tested the 0-context, 1-context, 2-context, and the extended-CPM algorithm on several circuits taken from the University of Michigan Aurora I CPU chip [3]. This chip was fabricated using the HGaAs II DCFL process from Vitesse Semiconductor Corp [17]. The circuits, listed in Table I, are taken from the control section of the CPU and vary in size from 8 to 184 gates. The gates in the circuits are all inverters and NOR gates with 2 to 6 inputs each. Because of the symmetry of these gates, the delay functions of all inputs are identical. The results in Table I show the critical long path delays, in picoseconds, obtained by the above four methods. These results are also compared with delays obtained form HSPICE simulations of the critical paths as identified by the extended-CPM algorithm. This was done so that the path delay comparison was not obscured by the possible presence of false paths or the effects of proximity.

[1] Note that for gates in level l, where $1 \leq l \leq c$, only $(l-1)$-context delays can be computed. The algorithm in Fig. 9 requires a slight modification to handle the first c levels in the circuit.

The 0-context delays were calculated according to a procedure described in the Vitesse Foundry Design Manual [17]. The delay equations for each gate type were determined, for rising and falling outputs, by assuming a fanin of 3 and a fanout of 1 and simulating the circuit with 0 and 300 fF loads in HSPICE. The gate delay was then obtained by linear interpolation through the resulting two data points, yielding the following *static* back-end delay equation

$$\Delta = \Delta_{\text{int}} + \alpha C \tag{18}$$

where Δ_{int} is an intrinsic delay, α is the rate of change of delay with load capacitance, and C is the total load capacitance. Note that there is no transition time information in this delay formula. For all the circuits in Table I, propagation delays for 0-context were calculated based on (18), and back-annotated into the circuit netlist. Next, a standard CPM-type timing analysis was performed to obtain the long path delays. The extended-CPM, 1-context, and 2-contex path delays were calculated using a *dynamic* gate delay model that was developed using the technique of dimensional analysis on those parameters that have a measurable effect on delay [9]. The equations that form the basis of the dynamic macromodel are as follows:

$$\frac{\Delta}{\tau_i} = f_\Delta\left(\frac{C}{W_E\tau_i}, \frac{W_L}{W_E}\right) \tag{19}$$

$$\frac{\tau_o}{\tau_i} = f_\tau\left(\frac{C}{W_E\tau_i}, \frac{W_L}{W_E}\right) \tag{20}$$

where C, W_E, and W_L are, respectively, the total load capacitance, the width of the switching transistor, and the sum of the widths of the load transistors. The functions f_Δ and f_τ are determined using a least-squares fit to experimental data obtained from circuit simulations. The accuracy of the macromodel as measured by the coefficient of multiple determination, R^2 [1], is better than 99.5% for all the gate types used and over a wide range of parameter values. In particular, this accuracy is guaranteed for input transition times ranging from 15 to 300 picoseconds.

Similar to the 0-context approach, the 1-context and 2-context delay techniques use the standard CPM timing analysis algorithm. The delay calculation method is different, however, and is based on the *Context-Delay* algorithm shown in Fig. 9. The characteristic time of GaAs DCFL used for context delay modeling was set equal to the transition time of a ring oscillator waveform (38 ps.)

Examination of the results leads to the following observations.

- The results obtained from the 0-context model are consistently less accurate than those obtained by the 1-context, 2-context, and extended-CPM models. This may be partly explained by the fact that the delay equation (18) incurs some error by ignoring the dc current drawn by the Schottky diodes on the fanout of the gate [10]. However, circuit simulations suggest that this error does not exceed 5%. The main source of inaccuracy seems to be the omission of the effect of signal transition time. This conclusion is supported by the strong correlation ($r = -0.84$) between the errors in the path delays and the *average* signal transition time along the critical path. Specifically, we can conclude that $71\%(r^2)$ of the error is accounted for by variations in the signal transition times.
- The path delays computed by the extended-CPM approach are within 1% of the HSPICE delays. This seems to confirm the accuracy of both the delay macromodel equations (19)–(20) and the path analysis procedure which takes signal transition times into account. The critical path delays computed by the front- and back-end approaches were in perfect agreement. This should not be surprising considering the symmetry of the NOR gates. However, there were minor differences between the two approaches on some subcritical paths. Careful examination of those paths revealed that the discrepancies were due to reconvergent fanout which results in signal proximity at some gate inputs. By tracing the cause of discrepancy of one of those paths, we discovered that a two-input NOR gate on the path had one input arriving at $A_1 = 499.3$ ps with a transition time $\tau_1 = 112.5$ ps and the second input arriving at $A_2 = 502.6$ ps with a transition time $\tau_2 = 90.5$ ps. Clearly, the two inputs overlap in this case, and our assumption about input proximity does not hold; fortunately, the path is subcritical. The departure time of the output using front-end delay calculation was calculated to be 589.5 ps, while for back-end, the departure time was 585.3 ps, which violates (17).
- The path delays computed by the 1-context approach fell within 5% of HSPICE delays. Again, there were no noticeable differences between front- and back-end models. This level of accuracy is consistent with the data shown in Fig. 8.
- The 2-context approach with front-end delay models provides accuracy comparable to the extended-CPM algorithm. This result indicates that it is possible to use standard path delay analysis tools and still get very good accuracy, provided that the gate delays are calculated using their 2-context. The back-end model results overestimate the delay in some cases by as much as 6.45% due to the way the back-end delay is calculated, which takes the maximum of the delays and output transition times associated with the different gate inputs.

V. Conclusions

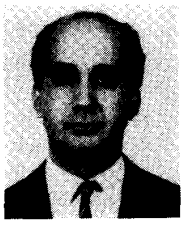
In this paper we have shown that failure to account for signal transition times in gate delay modeling can cause significant errors in path delay calculations. We proposed two approaches that account for signal transition times: an extension to the standard fixed-delay CPM algorithm which models signal transition time effects directly; and a context-based delay modeling step followed by standard fixed-delay CPM techniques. Both approaches have been shown to predict path delays with a high degree of accuracy. The context-based scheme offers the additional advantage of ease of integration with existing design tools. Signal proximity effects were neglected in both models: their impact on the accuracy of path delays and possible schemes for including them must await further research.

ACKNOWLEDGMENT

J. D. Wellman helped in the development of the DExtr program; his efforts are greatly appreciated.

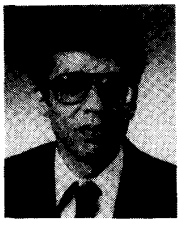
REFERENCES

[1] G. E. Box and N. R. Draper, *Empirical Model-Building and Response Surfaces.* New York: Wiley, 1987.
[2] M. A. Breuer, *Design Automation of Digital Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1972.
[3] R. B. Brown, P. Barker, A. Chandna, T. R. Huff, A. I. Kayssi, R. J. Lomax, T. N. Mudge, D. Nagle, K. A. Sakallah, P. J. Sherhart, R. Uhlig, and M. Upton, "GaAs RISC processors," in *Proc. IEEE GaAsIC Symp.*, Oct. 1992.
[4] H. C. Chen and D. H. Du, "Path sensitization in critical path problem," in *Proc. IEEE Int. Conf. Computer-Aided Design*, Nov. 1991, pp. 208–211.
[5] L. O .Chua, C. A. Desoer, and E. S. Kuh, *Linear and Nonlinear Circuits.* New York: McGraw-Hill, 1987.
[6] S. Devadas, K. Keutzer, and S. Malik, "Delay computation in combinational logic circuits: Theory and algorithms," in *Proc. IEEE Int. Conf. Computer-Aided Design*, Nov. 1991, pp. 176–179.
[7] D. A. Hodges and H. G. Jackson, *Analysis and Design of Digital Integrated Circuits.* New York: McGraw-Hill, 1983.
[8] N. P. Jouppi, "Timing analysis and performance improvement of MOS VLSI designs," *IEEE Trans. Comput.-Aided Design*, vol. CAD-4, pp. 650–665, July, 1987.
[9] A. I. Kayssi and K. A. Sakallah, "Delay macromodels for the timing analysis of GaAs DCFL," in *Proc. European Design Automation Conf.*, Sept. 1992.
[10] S. I. Long and S. E. Butner, *Gallium Arsenide Digital Integrated Circuit Design.* New York: McGraw-Hill, 1990.
[11] M. D. Matson and L. A. Glasser, "Macromodeling and optimization of digital MOS VLSI circuits," *IEEE Trans. Comput.-Aided Design*, vol. CAD-5, pp. 659–678, 1986.
[12] P. C. McGeer and R. K. Brayton, *Integrating Functional and Temporal Domains in Logic Design.* Kluwer, 1991.
[13] *HSPICE User's Manual*, Meta-Software, 1991.
[14] J. K. Ousterhout, "A switch-level timing verifier for digital MOS VLSI," *IEEE Trans. Comput.-Aided Design*, vol. CAD-4, pp. 336–349, 1985.
[15] D. T. Phillips, A. Ravindran, and J. J. Solberg, *Operations Research: Principles and Practice.* New York: Wiley, 1976.
[16] J. P. Silva, K. A. Sakallah, and L.M. Vidigal, "FPD—An environment for exact timing analysis," in *Proc. IEEE Conf. Computer-Aided Deisgn*, Nov. 1991.
[17] *Foundry Design Manual—Version 5.0*, Vitesse Semiconductor Corp., 1991.



Ayman I. Kayssi (S'89–M'93) received the B.E degree (with distinction) in electrical engineering from the American University of Beirut, Beirut, Lebanon, in 1987 and the M.S.E and Ph.D degrees, also in electrical engineering, from the University of Michigan, Ann Arbor, in 1989 and 1993, respectively.

Since January 1993, he has been at the American University of Beirut, as an Assistant Professor of Electrical Engineering. His research interests are in the area of computer-aided design of high-speed VLSI circuits and systems.



Karem A. Sakallah (S'76–M'81–SM'92) received the B.E. degree (with distinction) in electrical engineering from the American University of Beirut, Beirut, Lebanon, in 1975, and the M.S.E.E. and Ph.D. degrees in electrical and computer engineering from Carnegie-Mellon University, Pittsburgh, PA, in 1977 and 1983, respectively.

In 1981 he joined the Department of Electrical Engineering at CMU as a Visiting Assistant Professor. From 1982 to 1988 he was with the Semiconductor Engineering Computer-Aided Design Group at Digital Equipment Corporation in Hudson, MA, where he headed the Analysis and Simulation Advanced Development team. Since September 1988 he has been at the University of Michigan, Ann Arbor, as Associate Professor of Electrical Engineering and Computer Science. His research interests are primarily in the area of computer-aided design of integrated circuits and systems, with particular emphasis on numerical analysis, timing verification and optimal clocking, multilevel simulation, modeling, knowledge abstraction, and design environments.

Dr. Sakallah is a member of the Association for Computing Machinery.

Trevor N. Mudge (S'74–M'77–SM'84) received the B.S. degree in cybernetics from the University of Reading, England, in 1969, and the M.S. and Ph.D. degrees in computer science from the University of Illinois, Urbana, in 1973 and 1977, respectively.

While at the University of Illinois he participated in the design of several high-performance computers, and did research in computer architecture. Since 1977, he has been on the faculty of the University of Michigan, Ann Arbor, where he has taught classes on logic design, CAD, computer architecture, and programming languages. He is presently a Professor of Electrical Engineering and Computer Science and the Director of the Advanced Computer Architecture Laboratory—a group of eight faculty and sixty graduate research assistants. He is author of more than 100 papers on computer architecture, programming languages, VLSI design, and computer vision, and he holds a patent in computer aided design of VLSI circuits. His major research, at present, is the design and construction of a high-performance GaAs "microsuper-computer." In addition to his position as a faculty member, he is a consultant for several computer companies in the areas of architecture and CAD.

Dr. Mudge is currently Associate Editor for *ACM Computing Surveys*, Subject Area Editor for the *Journal of Parallel and Distributed Computing*, and a member of the Editorial board of *IEEE Parallel & Distributed Technology*. He is a member of the Association for Computing Machinery and the British Computer Society.