Static Timing Analysis Taking Crosstalk into Account[1]

Matthias Ringe
IBM Deutschland Entwicklung GmbH, Schönaicher Str. 220
71032 Böblingen; Germany
ringe@de.ibm.com

Thomas Lindenkreuz
Robert Bosch GmbH, 72703 Reutlingen, Germany

Erich Barke
Institute of Microelectronic Systems, University of Hanover
30167 Hanover, Germany

Abstract

Capacitance coupling can have a significant impact on gate delay in today's deep submicron circuits. In this paper we present a static timing analysis tool that calculates the longest path of synchronous circuits taking the impact of crosstalk on gate delays into account. We show that passive modeling of the coupling capacitance can significantly underestimate the delay and that an assumption of permanent worst-case coupling unnecessarily overestimates it. Our method is validated by comparison to Spice simulations.

1. Introduction

Due to the decrease in IC dimensions coupling effects become increasingly important. Apart from the functional impact [1][2], e.g. the generation of glitches, capacitance coupling can have a significant influence on gate delays [3] when adjacent wires switch in opposite direction. See Fig. 1 for an illustration of the problem.

In this paper we discuss the problem not only for one gate but for a complete synchronous circuit. Although coupling is a dynamic phenomenon our intention is to provide a timing analysis tool that can handle the delay impact of crosstalk statically. This is motivated by the fact that timing simulations cannot guarantee an upper bound for the delay since exhaustive simulation is impossible.

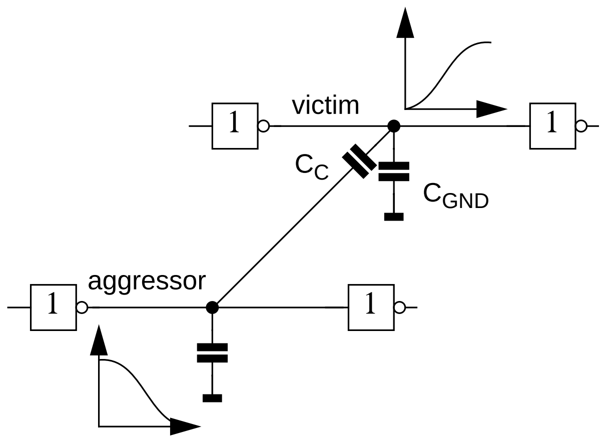

Fig. 1: Illustration of the problem

This article is organized as follows: At first, we discuss the gate model for delay calculation under coupling influence. Secondly, we describe our method of transistor-level static timing analysis. Subsequently, we present two algorithms that make use of the fact, that many lines are quiet during a clock cycle and therefore coupling does not occur. This lowers the upper delay bound of the longest path. Finally, we present layouted and extracted circuits to show the usefulness of our method. The comparison to Spice simulations of the longest path demonstrate the accuracy of our transistor-level static timing analysis.

[1] This work was supported by the German Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie under contract 01 M 3034. The authors are responsible for the contents of this publication.

2. The Coupling Delay Model

Although various delay models for classical delay calculation (see e.g. [4]) have been published few of them deal with the impact of coupling on gate delays.

The simplest and most often used approach is to double the value of the coupling capacitance and regard it as grounded. This is motivated by the fact that due to opposite switching the doubled charge flows through the coupling capacitance. This model ignores the active nature of coupling.

The coupling model presented in [3] describes gates as linear sources with constant resistance but variable slopes for the source voltages. The output situation is modeled by an effective capacitance which includes resistive shielding. Since all nonlinear effects are modeled linearly the cross coupling effect reduces to a problem of superposition of waveforms.

Although this model covers the non-passive nature of coupling there are some restrictions involved: The reduction to linear sources simplifies the behavior of MOS transistors, since in the saturation region they act as current sources whereas in the linear region they act as resistors. Another problem is that superimposing of waveforms means that they first have to be calculated. This is problematic since the actual aggressor waveforms depend on the coupling.

Spice simulations show that maximum delay is achieved when the aggressor voltage has a short ramp time. We get worst-case delay for an instantaneous voltage drop on the aggressor line. Another problem is *when* the coupling must occur to obtain worst-case delays. We suggest as a solution the following model, that consists of three steps. The discussion is made for a rising ramp on the victim line.

First, the aggressor line is quiet, i.e. the coupling capacitance is passive. When the victim voltage reaches a value of

$$V_{DD}\frac{C_C}{C_C + C_{GND}} + V_{th}$$

coupling occurs with an instantaneous voltage drop of V_{DD} on the aggressor line. The voltage on the victim line therefore drops to the value of V_{th}. After that drop the coupling capacitance gets and remains passive again. Note, that this model is a capacitive voltage divider.

For delay calculation the waveform before the occurrence of the coupling is completely ignored, i.e. the waveforms start with the value of V_{th}. This is motivated by the fact that the small glitch that is generated by coupling vanishes after one or two following gates. The impact of coupling then occurs only as an extra delay. It also keeps all waveforms monotonously rising or falling.

The natural choice of V_{th} as the threshold voltage of the transistors is not sufficient since it ignores the sub-threshold region. Certainly, a V_{th} that has no impact on the delay calculation has to be chosen. In our case the chosen value is 0.2 Volts while having a transistor threshold voltage of 0.6 Volts.

An advantage of this model is that the actual waveform on the aggressor line needs not to be calculated. It is sufficient to know whether or not there is activity. This makes the model particularly useful for static timing analysis.

A disadvantage of the model is that it is restricted to lumped capacitances. Wire delays are modeled by the widely used Elmore model. This model is known to overestimate the delay for long wires. In the worst-case sense this is acceptable. Note that in a timing-driven layout environment (e.g. [5]) the wire delays are not the major source of path delays.

3. The Gate Delay Model

Since our aim is to show the impact of coupling we chose a transistor-level approach for delay calculation to obtain best accuracy. As in [6] the DC behavior of the transistors is modeled by tables. Our approach differs in the iteration method since it uses the classical Newton approximation instead of the successive chord method proposed in [6]. Due to the fine discretization of the tables we do not get convergence problems. As shown in the results transistor-level timing analysis provides very accurate delay predictions compared to Spice.

4. Static Timing Analysis

Static timing analysis (STA) is a method to verify the timing of synchronous circuits. See e.g. [7][8][9][10] and references therein for a discussion of algorithms.

In short terms, the circuit is translated into a directed acyclic graph. The edges and vertices of the graph hold a delay. The task is to find the longest path through the graph which is usually done by a breadth-first-search (BFS). This algorithm allows to visit each vertex only once, i.e. it has linear complexity with respect to the number of edges and vertices. At each vertex only the worst-case waveform is propagated to the following stage. As mentioned in [10] static timing analysis is an exhaustive timing simulation during one clock cycle and therefore guarantees an upper bound for the delay of the complete synchronous circuit. Moreover, it also guarantees an upper delay bound for any event on each line. We make use of this fact later on.

5. STA and Capacitance Coupling

As mentioned above STA provides an upper time bound for the last event on each line. In other words, after this time the line is quiet to the end of the clock cycle. This leads to the following simple but efficient idea: We take into account if there can be an opposite transition on adjacent wires. If this is possible, we perform a waveform calculation according to the proposed coupling model. If not, we can regard the coupling capacitance connected to ground (or V_{DD}).

Assume a sample situation in which the aggressor line is quiet. The waveform on the victim line is to be calculated. To maintain the worst case we have to compare the latest activity on the aggressor line with the earliest possible activity of the current waveform on the victim line. Therefore, thresholds have to be defined. A safe and conservative choice is to take the same threshold voltages as chosen for the coupling model.

5.1 One Step Calculation

This algorithm is an extension to the normal BFS-algorithm. Since we have to compare the worst-case (last activity) on the aggressor line to the best-case on the victim line (first activity) we at first perform a calculation of the victim waveform assuming no coupling. This provides the lower time bound for the current waveform. Strictly speaking, this is not the lower bound since switching in the same direction may occur, but this is not within the scope of this discussion.

As an example we assume a rising transition on the output. The pseudo code is as follows:

```
w_bcs:=calculate waveform for best-case,
      i.e. all adjacent wires are quiet;
t_bcs:=time when w_bcs reaches $V_{th}$;
$t_{a,i}$:=time when adjacent wire i is quiet
      for falling transition;
foreach($t_a$,$t_{a,i}$)
   if(($t_a$ > t_bcs) or (line$_i$ is not calculated))
      $C_{coupling,i}$ is coupling capacitance
      according to the proposed model;
   else
      $C_{coupling,i}$ is a grounded capacitance;
w_wcs:=calculate_waveform();
insert w_wcs rising_event_queue of
      the victim line;
```

We stress that this algorithm still guarantees the worst-case delay. The experimental results show that it lowers the delay bound compared to a calculation in which permanent crosstalk is assumed.

This algorithm does not increase the complexity. The BFS is still performed in linear time. Compared to the normal BFS the waveform calculation is performed twice for each timing arc. It has the disadvantage that possibly the adjacent wires are not calculated and worst-case assumptions, i.e. coupling, about the activity on them must be taken. To overcome this restriction we propose the following extension.

5.2 Iterative calculation

The goal of this algorithm is to guarantee that a waveform calculation can be performed without the existence of uncalculated lines. Therefore, we call the one-step algorithm at least twice. After the first call (and any following call, too) the quiescent times are stored. From the second call on there are no more uncalculated adjacent wires. Hence, no worst-case assumptions about these wires must be made. The waveform calculation is iteratively refined. In pseudo code:

```
delay:=default;
do
   delay_old:=delay;
   delay:=do one-step sta;
   store quiescent times for each wire;
while(delay<delay_old);
```

The refinement is done at the expense of CPU time, of course. With no iterative improvement, a full STA is performed twice, with improvement it is performed at least three times. This algorithm can be sped up by using a method called *Esperance* introduced by Benkoski et. al. [11]. In this case only those wires that belong to long paths are recalculated.

Note, that this algorithm still guarantees an upper bound for the delay.

6. Experimental results

The following results are based on circuits of the ISCAS89 sequential benchmarks routed in a 0.5 µm process technology with two metal layers. The gates are sized and there is a clock buffer tree added. We compare:

1. Best case: All coupling capacitances are grounded with unchanged values, i.e. the effect of coupling is completely ignored. This is just a comparison value to emphasize the importance of taking coupling into account.
2. Static doubled: All coupling capacitances are grounded with doubled value. This is the classical approach to cope with the crosstalk impact on gate delays. Note, that permanent coupling of all adjacent wires is assumed.
3. Worst case: All cross capacitances couple according to the proposed model.

4. One step algorithm
5. Iterative algorithm

The following tables show the worst-case delay of the longest path of each circuit and the runtime of the STA program on a Sun Ultrasparc workstation.

Table 1: s35932 (17900 cells)

	Delay/ns STA	Delay/ns Spice	CPU sec. STA
Best case	13.62	13.47	168
Static doubled	14.70	14.53	168
Worst case	15.02	14.98	168
One step	14.63		317
Iterative	14.42		589

Table 2: s 38417 (23922 cells)

	Delay/ns STA	Delay/ns Spice	CPU sec. STA
Best case	24.73	24.67	195
Static doubled	26.89	26.80	195
Worst case	27.61	27.50	195
One step	26.93		358
Iterative	25.98		484

Table 3: s 38584 (20812 cells)

	Delay/ns STA	Delay/ns Spice	CPU sec. STA
Best case	25.03	25.20	222
Static doubled	26.94	27.05	222
Worst case	27.75	27.81	222
One step	27.05		443
Iterative	26.70		524

The best-case line compared to worst-case and the following new algorithms show that the delay impact of coupling certainly cannot be ignored.

Line two (static with doubled value) and three aim at the same goal, i.e. to provide a delay value when coupling is possible at any time. The difference between them is the model of coupling capacitances. The values show that the passive model obviously provides better results than ignoring coupling. However, the static model cannot cope with the active nature of coupling.

The two proposed algorithms lower the worst-case bound for the delay of the longest path making use of the fact that many lines are quiet most of the cycle time. The one-step calculation maintains the linear complexity of the breadth-first-search STA. The delay estimation is significantly more accurate. The iterative calculation improves the delay estimation further. Its main disadvantage is the increased runtime.

The Spice simulations of the longest paths were done with lumped resistances and capacitances extracted from the layout. Note the accuracy of the estimated delay values in comparison to the Spice simulations of the longest paths. Especially the worst-case coupling shows the accurate modeling of coupling since for the Spice runs piecewise linear sources had to be iteratively adjusted to obtain worst-case path delays at every coupling capacitance.

One could argue that a timing analysis with grounded coupling capacitances and doubled value provides almost the same delay values as the iterative refinement algorithm and therefore is a sufficient method. This assumption is wrong! The first method assumes permanent coupling on all wires but uses a model that does not necessarily provide the worst case [3]. The iterative refinement algorithm applies a more physical coupling model and exploits the fact that some lines cannot switch in opposite directions simultaneously.

Note that the impact of coupling is larger than the impact of wire resistance in these cases: The circuits s35932 and s38417 have a wire delay of about 0.2ns, the s38584 has a wire delay of 0.5ns. The impact of coupling is significantly larger (1.4ns, 2.8ns and 2.7ns, respectively).

7. Conclusions

A static timing analysis tool that can handle the effect of crosstalk on the delay of the longest path of synchronous circuits was presented. It makes use of a transistor-level waveform-based gate delay model. We demonstrated that ignoring or modeling coupling capacitances passively can lead to a significant underestimation of the delay even in a 0.5µ technology.

Due to the fact that many wires are quiet most of the cycle time, the assumption that all wires permanently couple can overestimate the delay. Therefore, we integrated two new algorithms that take into account if the adjacent wires can have an opposite transition during the transition on the current output.

The comparison to Spice simulations with extracted parasitics of the longest paths show the accuracy of our coupling model and the accuracy of transistor-level timing analysis in general.

To the best of our knowledge this is the first static timing analysis tool that takes the delay impact of coupling into account. Especially in comparison to Spice simulations the results demonstrate the importance of this effect.

References

[1] T. Stoehr, M. Alt, A. Hetzel, J. Koehl, "Analysis, Reduction and Avoidance of Crosstalk on VLSI Chips", *Proc. of the ISPD-98*, 1998

[2] A. Rubio, N Itazaki, X. Xu, K. Kinoshita, "An Approach to the Analysis and Detection of Crosstalk Faults in Digital VLSI Circuits", *IEEE T. o. CAD, Vol. 13, No. 3*, pp. 387-395, 1994

[3] F. Dartu, L. T. Pileggi, "Calculating Worst-Case Gate Delays Due to Dominant Capacitance Coupling", *Proc. of the 34th ACM/IEEE DAC*, pp. 46-51, 1997

[4] K. Eshraghian, N. H. Weste, "Principles of CMOS VLSI design", Addison-Wesley Publishing, 1993

[5] J. Koehl, U. Baur, T. Ludwig, B. Kick, T. Pflueger, "A Flat, Timing-Driven System for a High-Performance CMOS Processor Chipset", *Proc. of the DATE 98*, pp. 312-320, 1998

[6] F. Dartu, L. T. Pileggi, " TETA: Transistor-Level Engine for Timing Analysis", *Proc. of the 35th ACM/IEEE DAC*, pp. 595-598, 1998

[7] N.P. Jouppi, "Timing Analysis and Performance Improvement of MOS VLSI Designs", *IEEE T. o. CAD, Vol. 6, No. 4*, pp. 650-665, 1987

[8] T.G. Szymanski, "LEADOUT: A Static Timing Analyzer for MOS Circuits", *ICCAD-86 Digest of Technical Papers*, pp. 130-133, 1986

[9] D. E. Wallace, C. H. Séquin, "ATV: An Abstract Timing Verifier", *Proc. of the 25th ACM/IEEE DAC*, pp. 154-159, 1988

[10] J.K. Ousterhout, "A Switch-Level Timing Verifier for Digital MOS VLSI" *IEEE T. o. CAD*, Vol. 4, No. 3, pp. 336-349, 1985

[11] J. Benkoski, E. Meersch, L. Claesen, H. de Man, "Efficient algorithms for solving the false path problem in timing verification", *ICCAD-87 Digest of Technical Papers*, pp. 44-47, 1987