On the Chicken-and-Egg Problem of Determining the Effect of Crosstalk on Delay in Integrated Circuits [*]

Sachin S. Sapatnekar
ECE Department, University of Minnesota
200 Union Street SE, Minneapolis, MN 55455.
Ph: (612) 625 0025, Fax: (612) 625 4583, Email: sachin@ece.umn.edu

Abstract

This paper presents an approach to measuring of the effect of crosstalk on the delay of a net using an algorithm whose worst-case complexity is quadratic in the number of nets. The algorithm is amenable to being incorporated into the inner loop of a timing optimizer and is illustrated on a channel router, where it is seen to give improvements of about 20-30% in the average delay in a channel as compared to the worst case.

1 Introduction

Crosstalk is generally recognized as a major problem in IC design. This paper presents a novel approach to solving the chicken-and-egg problems related to the efficient measurement of the effect of crosstalk on the delay of a net using an algorithm whose worst-case complexity is quadratic in the number of nets, which is the best that can be achieved. Published techniques for crosstalk analysis typically work with either a very detailed and accurate analysis of the phenomenon (for example, [1,2]) or a very high-level model that captures the spirit, if not the details, of the crosstalk phenomenon (for example, [3–6]). The latter class of approaches has the advantage of speed over the former class, at the expense of accuracy, and has been therefore been used in the inner loop of optimizers. However, there is a need for greater accuracy without sacrificing the requirement of speed that is essential in the inner loop of an optimizer.

The goal of this work is to develop a technique that is intermediate to the two in accuracy and speed, and to show its application to optimal crosstalk-conscious channel routing. The delay calculation procedure uses the Elmore delay model in the examples shown here, but the assumptions used herein are general enough that the only requirement of the delay model is that it should show an increased delay for an increased capacitance. Therefore, a higher order AWE-like delay model [7] is equally applicable to this basic framework. The application of this approach to optimal channel routing is shown, and it may be applied to other optimization scenarios.

2 Effect of Crosstalk on Timing

This work models a wire as a succession of π RC segments connected in series. We assume that the widths, w_i, of the wires are kept constant through the analysis and optimization. The resistance, R_i, and intrinsic capacitance, C_i, of the i^{th} segment are given by the formulæ$R_i = \alpha\ l_i/w_i$ and $C_i = \beta\ l_i\ w_i$, where w_i and l_i are, respectively, the width and length of the i^{th} segment, and α and β are constants of proportionality for the resistance and intrinsic capacitance (including the fringing capacitance), respectively. The coupling capacitance, C_c, between two adjacent nets is given by $C_c = \gamma\ \text{overlap}_i$, where overlap_i is the length along which the nets run next to each other.

The role of the coupling capacitances is greatly dependent on the relative switching times of the nets [8]:
- If one net switches and the other remains inactive, then the equivalent coupling capacitance $C_{c,eq} = C_c$.
- If both nets switch at the same time in opposite directions (i.e., $1 \rightarrow 0$ and $0 \rightarrow 1$), then $C_{c,eq} = 2C_c$.
- If both nets switch at the same time in the same direction, then $C_{c,eq} = 0$.

The complexity of this relationship arises from the interrelationships between the timing behavior and the coupling capacitance. The value of the equivalent coupling capacitance is affected by the switching time, which, in turn, is affected by the value of the coupling capacitance.

To elaborate on this, consider two wires that are laid out adjacent to each other. If the input signals to driver of the two wires switch between times $(T_{min,1}, T_{max,1})$ and $(T_{min,2}, T_{max,2})$, respectively, and if the delays required to propagate the signal along the wires are in the range $(d_{1,min}, d_{1,max})$ and $(d_{2,min}, d_{2,max})$, respectively, then the intervals during which the lines switch are $(T_{min,1} + d_{1,min}, T_{max,1} + d_{1,max})$ and $(T_{min,2} + d_{2,min}, T_{max,2} + d_{2,max})$, respectively. Therefore, the following relationship holds between the switching times and the equivalent coupling capacitance, $C_{c,eq}$ (note that it is possible for some of the above intervals to be empty):

[*]This research was supported in part by the SRC under contract 98-DJ-609 and by the NSF under award CCR-9800992.

Table 1: Variation of $C_{c,eq}$ with switching time.

Interval	$C_{c,eq}$
$(\max\{T_{min,1}+d_{1,min}, T_{min,2}+d_{2,min}\}, \min\{T_{max,1}+d_{1,max}, T_{max,2}+d_{2,max}\})$	0 or $2C_c$
$(\min\{T_{min,1}+d_{1,min}, T_{min,2}+d_{2,min}\}, \max\{T_{min,1}+d_{1,min}, T_{min,2}+d_{2,min}\})$	C_c
$(\min\{T_{max,1}+d_{1,max}, T_{max,2}+d_{2,max}\}, \max\{T_{max,1}+d_{1,max}, T_{max,2}+d_{2,max}\})$	C_c

The chicken-and-egg problem arises from the fact that $d_{1,min}$, $d_{1,max}$, $d_{2,min}$ and $d_{2,max}$ are dependent on the value of $C_{c,eq}$, which is itself dependent on the values of $d_{i,min}$ and $d_{i,max}$, $i = 1, 2$. Therefore, an iterative approach is required.

It should be pointed out here that the use of the three values of 0, C_c and $2C_c$ is an approximation. In reality, depending on the switching time and the transition time of the two waveforms, the equivalent coupling capacitance does not have to be restricted to these three values. This algorithmic framework can easily be extended to handle such cases, provided the effective coupling capacitances (e.g. $5C_c$) are provided for each transition.

3 An Algorithm for Correct Crosstalk Estimation

We define a *spatial adjacency graph*, G_s, whose vertices correspond to the k nets in the channel. An edge is drawn between vertices i and j if the spans of nets N_i and N_j intersect. If two vertices are connected by an edge, the corresponding nets will affect each other by means of a coupling capacitance if they are placed adjacently.

All transitions are assumed to be sharp, and occur at a time given by the delay; we happen to use the Elmore delay model here, but the basic approach may be extended to any other model.

3.1 Outline of the Algorithm

For each driver of a net, a switching interval (T_{min}, T_{max}) and a source resistance, R_d, is specified. The goal of the algorithm is to incorporate the information in the G_s graph and the adjacency information derived from the channel routing solution to arrive at a range (T_{start}, T_{end}) for all of the wires in the channel.

We define the self-delay, d_s, of a line as its RC delay calculated by considering only the intrinsic capacitance of the line. Note that the self-delay is calculated without incorporating the effects of coupling capacitance; consideration of the coupling capacitance can only cause the delay to increase, and hence the self-delay is a lower bound on the delay of the line. The task of this algorithm is to determine whether the correction due to coupling capacitance should assume a capacitance of C_c or $2C_c$ [0 or C_c] for the maximum [minimum] switching time. As a notational convenience, let delay(C_c) be the delay on the line due to the coupling capacitance of C_c for each neighbor of a given wire. The initial switching interval is set to the value of (T_{start}, T_{end}), where $T_{start} = T_{min} + d_s$ and $T_{end} = T_{max} + d_s +$ delay(C_c), both of which are clearly lower bounds on the earliest and latest switching times for the wire. The pseudocode for the algorithm is shown below:

```
ALGORITHM Update_Switching_Times
1.   For each net
2.       calculate its $d_s$ and delay($C_c$) and update its $T_{start}$ and $T_{end}$ value
     /* OUTER LOOP */
3.   Repeat
         /* FORWARD PASS: Update the latest switching time for each net using
            $C_{c,eq}$ = $C_c$ or 2 $C_c$, as appropriate */
4.       Repeat
5.          For each net $i$
6.             For each neighbor $j$ of $i$ in $G_s$
7.                Update $T_{end}$ for $i$
8.             For each neighbor $j$ of $i$ in $G_s$
9.                Update $T_{end}$ for $j$
10.      until (no $T_{end}$ changes)
         /* BACKWARD PASS: Update the earliest switching time for each net using
            $C_{c,eq}$ = 0 or $C_c$, as appropriate */
11.      Repeat
```

```
12.        For each net $i$
13.           For each neighbor $j$ of $i$ in $G_s$
14.              Update $T_{start}$ for $i$
15.           For each neighbor $j$ of $i$ in $G_s$
16.              Update $T_{start}$ for $j$
17.     until (no $T_{start}$ changes)
18.  until (no $T_{end}$ or $T_{start}$ changes)
```

In practice, the changes in the forward and backward passes are only made for neighbors of nets that were altered in the previous iteration, except in the first iteration of the outer loop, where all nets are processed. The neighbors of a wire j above correspond to adjacent vertices in the G_s graph. The updates in each pass are performed using the scheme in Table 1, with the difference that the wire delays are calculated using the values of $C_{c,eq}$ based on the current values of T_{start} and T_{end} for the nets. The update formulæ are as follows:

T_{end} **updates**

- If $T_{end}(j) > T_{end}(i) > T_{start}(j)$, then set $T_{end}(i) = \text{Update}(T_{end}(i), 2C_c)$.
- If $T_{end}(i) > T_{end}(j) > T_{start}(i)$, then update $T_{end}(i) = \max[\text{Update}(T_{end}(j), 2C_c), T_{end}(i)]$.
- If the two intervals do not overlap spatially, $T_{end}(i) = \text{Update}(T_{end}(i), C_c)$ and $T_{end}(j) = \text{Update}(T_{end}(j), C_c)$.

T_{start} **updates**

- If $T_{start}(j) > T_{start}(i)$, then update $T_{start}(i) = \min[\text{Update}(T_{start}(i), C_c), T_{start}(j)]$.
- If $T_{start}(i) > T_{end}(j)$, then update $T_{start}(i) = \text{Update}(T_{start}(i), C_c)$.
- If $T_{end}(j) > T_{start}(i) > T_{start}(j)$, then $T_{start}(i)$ is left unchanged.

Several speedup heuristics have been developed, but are not described due to space limitations.

3.2 Theoretical Results and Complexity

Some theoretical results are stated below; the proofs are omitted due to space limitations.

Theorem 1: The iterative procedure in Algorithm Update_Switching_Times converges.

Theorem 2: The computational complexity of the algorithm is $O(nm^2 + mn^2)$, where n is the number of nets and $m < n$ is the maximum number of nets that are spatially adjacent to any net. Therefore, assuming that m is bounded by a constant, the complexity of the procedure is $O(n^2)$.

Comment 1: In practice, this upper bound was never seen to be reached and is unlikely to be reached for problems where the wires are evenly distributed. However, it can be illustrated to occur in a pathological case where a set of n bus lines run next to each other. In such a case, each line is capable of affecting every other line, and any algorithm must incorporate the $O(n^2)$ interactions.

Comment 2: The proof is valid for any delay model that satisfies the requirement that as the load capacitance increases, the delay will increase. This is a requirement satisfied by most models in an IC environment.

4 Experimental Results

The algorithm to minimize the objective function by reordering, subject to horizontal and vertical constraints, was implemented in C and executed on a Sparc Ultra 1/170 workstation. In our implementation, we assumed that the rise times are equal to the fall times, but this is not essential, and the procedure can be extended easily to handle rise and fall transitions separately.

A summary of the results is shown in Table 2 for 0.25μm technology parameters. The algorithm was used to reorder eight different examples, keeping the number of tracks the same as that in the original solution that was obtained from a Yoshimura and Kuh channel router [9] that optimizes the height of the channel. The eight examples are taken from [9], with the last example being the routing of the Deutsch difficult example with doglegs.

The second column of Table 2 shows the number of nets for each example. The third column shows the improvement in the objective function at the end of the simulated annealing run, as compared to the objective function value in the original channel. It is seen that improvements of up to 28% over the initial configuration are achievable. The switching times of the drivers were chosen randomly, and it should be pointed out that the extent of the improvement depends on the switching times of the drivers, and there will be situations where significantly larger improvements may be possible. However, the intent of this table is to illustrate that the reordering process

Table 2: Results of Channel Reordering on Timing

	Number of nets	Improvement over initial	CPU Time	Improvement over worst
ykfig2	10	0.9%	12.2s	0.9%
yk1	21	10.7%	34.3s	20.6%
yk3a	45	7.4%	88.8s	13.7%
yk3b	47	11.9%	87.5s	24.2%
yk3c	54	28.2%	107.8s	38.4%
yk4b	54	17.9%	115.1s	25.0%
yk5	60	14.5%	118.6s	26.6%
Deutsch	72	10.7%	352.3s	22.1%

provides a significant advantage even for randomly chosen timing intervals. The CPU times for the run are shown in the next column.

To obtain an idea of how much the optimal solution differs from the worst solution, the simulated annealing algorithm was executed again, this time with the objective of *maximizing* the objective function. At the end of this run, we have a reordered channel where the effects of crosstalk correspond to the worst possible scenario. The difference between this objective function value and the objective function value obtained earlier provides an idea of how much improvement is possible between the most optimal and the least optimal channel routing solution. Note that both of these solutions are valid solutions with the same number of tracks, and it is quite possible for a CAD tool that is not crosstalk-conscious to come up with the worst-case solution. It is seen that the difference between the best case and the worst case can be as large as 38%.

References

[1] D. H. Xie and M. Nakhla, "Delay and crosstalk simulation of high-speed VLSI interconnects with nonlinear terminations," *IEEE Transactions on Computer-Aided Design*, vol. 12, pp. 1798–1811, Nov. 1993.

[2] P. D. Gross, R. Arunachalam, K. Rajagopal, and L. T. Pileggi, "Determination of worst-case aggressor alignment for delay calculation," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 212–219, 1998.

[3] T. Gao and C. L. Liu, "Minimum crosstalk channel routing," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 692–696, 1993.

[4] K. Chaudhary, A. Onozawa, and E. S. Kuh, "A spacing algorithm for performance enhancement and cross-talk reduction," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 697–702, 1993.

[5] D. A. Kirkpatrick and A. L. Sangiovanni-Vincentelli, "Techniques for crosstalk avoidance in the physical design of high-performance digital systems," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 616–619, 1994.

[6] T. Xue, E. S. Kuh, and D. Wang, "Post global routing crosstalk risk estimation and reduction," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 302–309, 1996.

[7] L. T. Pillage and R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Transactions on Computer-Aided Design*, vol. 9, pp. 352–366, Apr. 1990.

[8] L. Gal, "On-chip cross talk - the new signal integrity challenge," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 251–254, 1995.

[9] T. Yoshimura and E. S. Kuh, "Efficient algorithms for channel routing," *IEEE Transactions on Computer-Aided Design*, vol. CAD-1, pp. 25–35, Jan. 1982.