A ROADMAP OF CAD TOOL CHANGES FOR SUB-MICRON INTERCONNECT PROBLEMS

Lou Scheffer
1 Cadence Design Systems, lou@cadence.com

ABSTRACT
Correct interconnect design is crucial to making sub-micron chips that work. Right now CAD tools do little or nothing to help prevent signal integrity and reliability problems in the interconnect. However, they need to address this problem very soon since a significant percentage of the nets in a 0.25 micron process will be affected by one or more signal integrity problems. These must be fixed automatically, since there will be too many to be fixed by hand. This paper looks at some possible solutions to this problem. There are two basic ways to attack this problem - methodology changes and CAD tool changes. Neither is likely to solve the problem alone, and an optimum solution will include some of each. This paper concentrates on the CAD tool changes needed to automatically address sub-micron interconnect issues. A large number of tools in the design flow must be changed, and the data flow between these tools will be considerably more complex than in existing flows.

1. INTRODUCTION - WHAT ARE THE PROBLEMS?
The main interconnect problems are signal integrity (in particular, crosstalk noise margins), accurate delay calculations, and electromigration. The noise margin of a signal is the amount of noise that can be tolerated on that signal before a malfunction occurs. Exceeding the noise margin will cause a chip to malfunction under certain, normally rare, cases. In theory, this problem can be detected by testing if the right test vector set can somehow be found, but finding the right test vector set is practically impossible. No permanent damage to the chip occurs as a result of a crosstalk error, just a wrong result on one particular clock cycle.

Accurate delays are a problem since a big fraction of the delay depends on the effective capacitance as seen by the driving cell. This capacitance, however, depends heavily on the logic and timing behavior of the neighbors of the interconnect.

Electromigration is a long term reliability problem. If too much current goes through a given line, the line may fail as the metal atoms migrate from the flow of current. Such a failure is permanent but slow to develop. It cannot be found by testing when the chip is new - it is a problem that develops over the life of a chip. All of these problems become worse as design rules shrink further into the sub-micron realm. The primary culprit is the finer geometry interconnect wires - as the wires shrink the resistance goes up faster than the capacitance goes down. Furthermore, modern processes with fine lines and many interconnect layers have a higher ratio of coupling capacitance to grounded capacitance. Finally, faster clock rates increase the AC current on signal lines. All of these effects contribute to the crosstalk, delay calculation, and electromigration problems.

2. NOISE MARGIN
For a chip to function correctly, the noise margin of each signal must account for several sources of noise:

- Cross-coupling. When an adjacent signal switches, it induces noise into the desired signal. This problem is worse in sub-micron processes, since finer geometries and numerous metal layers mean that more of the capacitance is signal to signal, not signal to ground.
- Power supply noise. Two gates powered by different supplies have their noise margin reduced by the amount of non-common power supply noise between them.
- Thermal differences. If different parts of a chip are at different temperatures, and different temperatures result in gate threshold differences, then there is a thermal component to the noise budget. This is important for processes with temperature sensitive thresholds (bipolar processes) or those with poor thermal conductivity substrates (GaAs).
- Statistical differences. In a sub-micron process, there are no longer a statistically huge number of impurity atoms under each gate. As a result, there are statistical variations in the thresholds of the gates. There may also be cross-chip gradients, but these are usually small and primarily of interest to analog designers.

Thermal and statistical differences are small and usually ignored, and we will follow that practice here.

2.1. Electromigration
In the most popular technology, CMOS, almost all of the currents in the chip are dynamic, so electromigration and supply drop issues cannot be addressed without considering the activity of the circuit (how often, on the average, each node switches). Furthermore, the electromigration concerns are different for power supply nets and signal nets.
Figure 1. Real parasitics are distributed.

- Power supply nets - The width of power supply and ground wires depend on the current expected to flow through them. There is both an AC and a DC component to this current.
- Signal nets - Each signal has an expected AC current from charging and discharging the node. If this AC current exceeds the capacity of the wire, a larger wire must be used, or the net divided into smaller pieces. For CMOS technology, there is no DC component to the current in a signal net.

2.2. Analysis of the crosstalk and delay problems

Determining the exact amount of crosstalk, or the exact timing of a circuit, is a very difficult problem. This is because the timing and slopes of every signal depends heavily on the effective capacitance as seen by the net driver. This, in turn depends on the delays, slopes, and logic states of the neighboring nets. The logic states, in particular, vary considerably from one cycle to the next. Furthermore, the behavior of the neighboring nets, in turn, depends on their neighbors, including the original net. Because of these dependencies, in a sub-micron design the timing or slope of a signal is not a single number - it is a range, and varies from cycle to cycle, depending on the signals on the adjacent nets. Finding an consistent solution showing exactly how big the range is for each signal is very difficult (in fact it’s NP-complete - see appendix A.)

Looking at the equivalent circuit of a net subject to crosstalk, as seen in figure 1 above, we see that the parasitics are distributed. If we let $C_g$ be the sum of all the grounded caps, and $C_c$ be the sum of all the coupling caps, then (assuming all signal swings are equal) the effective capacitance as seen by the driver can be as high as $C_g + 2C_c$, if all the interfering nets are charging in the opposite direction to the signal under consideration. The effective capacitance could be as low as $C_g$. (Even this is a simplification - these $C_s$ are integrated over the whole transition, which is not quite right either - it’s actually the portion of the transition up to threshold that counts.) These are not small effects - if $C_c = C_g$, then the effective capacitance, cell delays, and slopes can cover a 3:1 range for a single gate, just depending on the neighbors behavior. This variation occurs not from chip to chip but from clock cycle to clock cycle on the same chip!

These are not just theoretical concerns - the exact situation outlined above can happen if a bus is routed in adjacent tracks. Consider the situation where (for example) bit n is rising. On one cycle, adjacent bits $n-1$ and $n+1$ are falling. The transitions on a bus may well happen at the same time, so the driver for bit n sees $C_g + 2C_c$. The next time bit n rises, perhaps bits $n-1$ and $n+1$ are rising, so the driver for bit n sees only $C_g$. If $C_g$ and $C_c$ are comparable, which is realistic, then the same driver, driving the same net, gives a factor of 3 different risetimes and delays on two different cycles.

3. APPROXIMATIONS

Although NP-completeness means we are unlikely to solve the full problem, we can solve enough of the problem to be useful for making chips. To do this, we must ensure that the crosstalk will not create logic problems, and that the worst case timing is accounted for. To do this, we create worst case estimates for crosstalk and timing, and if these estimates exceed safe limits, we fix the circuit even though it might not really need it. Because the estimates are worst case, we will always get a working circuit, but we may spend more resources than necessary. The better the estimates, the less resources we will waste on unnecessary shielding and buffering.

The best approach so far for obtaining these worst case estimates seems to be to assume that any pair of signals may interact unless we can prove otherwise. Proof of non-interaction may be:

- Physical - in digital logic, if the signals are never adjacent then the interaction can in general be ignored. This applies to both crosstalk and timing.
- Electrical - if we know the cross coupling capacitances and wire and driver resistances, we may be able to prove that we do not have a crosstalk problem, and we can set tighter limits on the timing variation caused by crosstalk.
- Temporal - if the two signals cannot switch at the same time, then they have no effect on each other’s timing. Crosstalk must still be evaluated, though.
- Logical - Based upon the logic function of the circuit, we may be able to prove that two signals cannot switch at the same time. Fitzpatrick and Sangiovanni-Vincentelli have studied this [1]; they conclude that in general relatively few interactions can be ruled out on purely logical grounds. An important exception is state machines with sparse transition graphs, where a significant number of interactions can be ruled out.

3.1. Upper bounds on crosstalk

One way to create an upper bound on crosstalk is as follows: First, perform one pass of slope propagation assuming that every signal is loaded only by $C_g$. This will give fast slew rates for all signals (gates have fast slewing inputs, and all cross coupling is assumed to be in the favorable direction.)

Second, rather than handling the full distributed case, we make the simplifying assumption seen in figure 2 (which is analytically tractable - we can calculate the exact crosstalk at the output for a known ramp on the interfering signal). We assume a worst case topology - all grounded capacitance is near the driver, all the coupling capacitance is near the load, and all the wire resistance is between them. Finally, since the system is linear, we can compute the total crosstalk bound as the sum of each crosstalk induced by each signal individually.

The worst cases as computed above are not overly pessimistic - they can and do happen on real chips. In a bus,
for example, all the neighbors of a signal may switch on the same stimulus, and may all move in the same best or worst case direction. Another example is a undriven net that is being counted on to hold a value. If all the coupled nets move in the same direction (not necessarily at the same time) then the above procedure estimates the crosstalk accurately.

4. POSSIBLE SOLUTIONS TO THE SUB-MICRON SIGNAL INTEGRITY PROBLEM

There are two basic approaches to solving signal integrity problems - changing the CAD tools, and changing the design methodology. Changing the existing tools, or adding new tools to the flow, is one approach. If long lines have crosstalk problems, for example, then a new tool that converts long lines to short ones by inserting buffers will solve this problem.

A different approach is through methodology. These approaches restrict the design space, and utilize design resources, to reduce the risk of signal integrity problems. For example, to reduce crosstalk problems, we could assign every third track on each layer to a power supply or ground wire, or route signals as differential or twisted pairs, or use ground planes. These techniques reduce the amount of crosstalk on signals at the cost of routing resources. Alternatively, we could restrict the use of circuits that are unusually sensitive to crosstalk. For example, dynamic logic could be confined to the interior of pre-characterized cells. Finally, we can use methodology to make CAD problems easier. For example, inductance calculations are very difficult if the return current path is not well defined. Adding power and ground wires not only reduces the inductance but makes it easier to compute as well.

All methodology approaches outlaw certain practices (which might be OK in some cases) in the interest of making the design process simpler. As usual for these complex problems, probably the best overall solution will be a combination of methodology and CAD tool improvements.

Here are some examples of the type of rule sets and tools that we might use to rule out sub-micron interconnect problems. Both more complex and simpler rule sets are possible. We'll start with a simple set of rules to verify a routed chip:

- For each input, define a noise allowance. This is the maximum amount of noise we are willing to tolerate. This will be different for different nets and circuits; a dynamic node can tolerate almost no excursions above threshold, while static CMOS logic is much more forgiving. These can probably be entered as properties on the inputs of gates in a library.

- Divide the total noise margin into two components, one for ground noise and once for crosstalk noise. (An example of a methodology rule.)

- Define a function for electromigration limits - given the amount of AC and DC current in a net, this function computes the minimum width of the wire needed to carry this current for the life of the chip.

Next, we need three new tools.

- A crosstalk checker that examines the results of the extractor, computes an approximation of the crosstalk as explained above, compares it to the crosstalk limit, and reports any errors.

- A power and ground checker that uses the activities and capacitances to compute currents and voltage drops, and compares them against the limits.

- An electromigration checking program that compares the width of each wire with the limit computed from its current, and reports any errors.

A chip that passes these tests should work - (although the converse is not true - a chip could fail these tests and still work - for example, there might be too much power supply noise, but if it’s correlated at the source and the receiver the chip would still work.)

In the long run, however, the goal is not finding possible problems, it’s building chips that do not contain such problems. The rules above aren’t enough to do this conveniently since the data that are needed for this calculation (coupling capacitances and wire resistances) are not available during some earlier steps (such as placement). Therefore we may want to establish a few more rules to make building designs that are correct by construction easier. An example might be:

- Define minimum and maximum acceptable slew rates for on-chip signals. The maximum slew rate limit allows us simplify crosstalk and electromigration rules; the minimum slew rate limit prevents excessive power dissipation.

This helps because given slew rate limits and physical parameters (coupling capacitance per unit length), we can now compute a safe length for a net that depends on the driver type. If the net is shorter than this limit, it can be guaranteed to have no crosstalk problems. This length limit is something that a placement tool can easily use.

So a practical crosstalk analysis/synthesis system might allow any of the following criteria to indicate that a net is OK:

1. The net is sufficiently short (depending on the driver type).

2. The net has an adequate crosstalk margin. This is a more detailed calculation, using the $C_c$, the $C_g$, $R_s$, $R_w$, the slew rates of the interfering signals, and the analytic solution from above.

3. The net is a power supply or ground net.

4. The net has been marked ‘don’t care’ or ‘already analyzed’ by the designer.

5. CAD TOOL CHANGES TO SOLVE SIGNAL INTEGRITY PROBLEMS

So what would a CAD system look like that could design chips free of crosstalk and electromigration problems? First, we need to define formats for specifying the constraints needed to create a working chip, and the data
needed to compute whether the constraints are met. Note that the data, and the constraints, may be process specific, library specific, or design specific. Examples of each of these constraints and data are show in Table 1, below.

**Table 1 - Examples of data and formats**

<table>
<thead>
<tr>
<th>Data</th>
<th>Design</th>
<th>Library</th>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transition</td>
<td>Transition</td>
<td>power(l,c)</td>
<td>cap/mm</td>
</tr>
<tr>
<td>counts per</td>
<td>equivalent</td>
<td>per cell</td>
<td>of wire</td>
</tr>
<tr>
<td>net</td>
<td>VCD or</td>
<td>DCL or</td>
<td>??</td>
</tr>
<tr>
<td></td>
<td>equivalent</td>
<td>equivalent</td>
<td></td>
</tr>
<tr>
<td>Constraints</td>
<td>false paths</td>
<td>noise</td>
<td>Wire</td>
</tr>
<tr>
<td></td>
<td></td>
<td>allowed at</td>
<td>width(ldc,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>input</td>
<td>1xc)</td>
</tr>
<tr>
<td>Format</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCF or</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>equivalent</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Next, we need to modify CAD programs to respect these limits. Using the data, they need to see if the constraints are met. The CAD tools must at least report violations, and fix them automatically where possible. Once a CAD tool has identified a signal integrity problem, there are usually several ways to fix it. For example, a crosstalk problem on a single driver net can be fixed by inserting buffers until rule (1) applies. Multiple driver nets are harder to correct automatically. Some things that help to meet rule (2) are: widening the wire, leaving adjacent tracks blank or filling adjacent tracks with grounded signals.

DC electromigration problems and power supply voltage drop problems must be fixed by widening the power supply wires or adding more of them. A much more difficult alternative is to redesign the sub-system to use less power.

A crosstalk problem can be fixed by widening the wire (up to a point). After that the net must be subdivided, which reduces the current in each section.

Here are some of the CAD programs, tools, and methodologies that need to be modified to make this happen.

5.1. Define the constraint contents and format
CAD vendors and users need to work together to define a methodology and constraints that will ensure that chips will work, be relatively simple to check, and not give up too much performance. This will be difficult since there will be considerable disagreement on the appropriate constraints. The high performance designers will want to trade off among all the possible variables (for example, if two gates share a nearly common power supply, then allow more crosstalk), while the ASIC customers will want a simple scheme that is very predictable.

5.2. Data and constraint formats
Data and constraints for libraries, technologies, and designs are needed to address signal integrity (see Table 1 above.) Unfortunately, only a few of these have well accepted common formats. Standardizing these formats would be an excellent topic for Sematech, VSI (Virtual Socket Interface) or some other multi-vendor, multi-user forum to discuss.

Until recently, design constraints have been expressed in tool specific form (a common example would be timing constraints expressed as commands to a synthesis tool.) Unfortunately, these are too specific and subject to change with new versions of the tools. Another possibility, called GCF (for General Constraint Format), is designed to solve this exact problem. This is a new format, under consideration by OVI, designed to express the constraints needed for chip design.

Design specific data can probably be handled by existing formats: cross coupling between nets can be expressed in formats such as DSPF (Detailed Standard Parasitic Format), signal transition information in VCD (Verilog Change Dump), and so on. These formats may contain more data than is required, however, and a new format(s) might be preferable for efficiency reasons.

The library data and constraints (power consumption and input noise allowed) will probably be expressed in an extension to library timing formats. OVI has endorsed DCL (Delay Calculation Language) here, but a more common format is a timing library for synthesis tools.

Technology data and constraints have no standard representation so far.

5.3. Changes to delay calculators/static timing analyzers
The delay calculator and the timing analyzer need to interact very closely. Rs and Cs cannot be accurately reduced to delays until the simultaneous switching behavior of signals is known. But computing whether simultaneous switching is possible is the job of the timing analyzer, which requires that the Rs and Cs have already been reduced to delays!

One possible solution is to have the reduction and timing analysis iterate. The first cut at reduction could produce a range of possible delays. These would propagate through the timing verifier, giving ranges of switching times at the outputs. Then we can reduce again, using the new knowledge of which signals can simultaneously switch. Since the ranges are always reduced, and bounded below by zero, convergence is guaranteed. The speed of this convergence, however, is not yet known.

In addition, timing verifier/delay calculator combinations might want to implement the following changes:

- Generate a file that lists the slope for every signal. This will be needed for computing power dissipation, which is needed for power supply wire sizing.
- Generate an output file that lists, for every pin, what the predecessor pin was used when computing the slope. This amount a graph over the design for computing slopes. This way tools such as post-placement synthesis (PPS) won’t have to have complex backtracking algorithms. Naturally, if a lot of changes are made there is a good chance that this file will not be 100% accurate, but it’s a lot better than nothing.

5.4. Changes to analog analysis tools and libraries
Noise limits are most usefully computed by analog simulations. This is a one time setup that is done once per process or per library. These include:

- For each cell in a library, compute the allowable noise on each input.
- For typical wiring configurations, find the total coupling and grounded capacitance per unit length. This might be done as follows:
  1. Set up worst case wiring configurations, perhaps a 'victim' net with 'aggressor' nets on both sides and top and bottom.
  2. Drive nets of different lengths with drivers of different types, and look at the resulting crosstalk.
  3. Automatically generate the effective values for Cg and Cc from these results.
- We need to add a energy per transition field for each output and input in the library. (Inputs are needed because on some cells a transition on the input will cause dissipation even if the output does not switch.) This might be expressed as a total energy per transition based upon output load and input slew. This could be a two dimensional table of the form already used for synthesis timing. This information will be combined with the transition counts above to calculate the power drawn by each section of the chip, which in turn will be used to correctly size the power lines.
- CAD vendors could sell a standardized service of computing these rules.

5.5. Changes to physical verification
- We need to compute how much of the capacitance of each signal is to ground, and how much is to other signal nets. It’s much better if we know which nets, for otherwise we will have to assume the worst in terms of slew rates. We might want to consider some basic timing as well, such as the clock phase of the extracted signals. If two adjacent signals change on different phases, the capacitance between them can be regarded as grounded. (This can also be done in the delay reducer/timing analyzer if it is not done here.)
- We may need inductance for some on-chip signals. See Appendix B.
- We might want a stand alone tool to check for interconnect problems. This might be run after placement, after PPS (see below), and after final routing. Another possibility is a common sub-routine library that is called by all these tools.

5.6. Changes to placement tools
Ideally, placers could understand all signal integrity rules and apply them during placement. In practice this is difficult for at least three reasons:
- Some of the information needed to determine if a problem exists is not available at placement time (for example, signal adjacencies).
- The complete signal integrity rules are quite complex, and relatively slow to compute. The placer needs to consider a huge number of possible configurations, and cannot spend a lot of time evaluating the design of each net.
- Applying the full signal integrity rules complicates the flow since there are currently no provisions for modifying the netlist in the placement step.

These complications make it hard for a placer to do all the work required to meet the rules. However, one change that can easily be made is to change the wire delay evaluation in timing driven placement to take into account buffering that will be inserted later to address crosstalk and delay problems.

5.7. Changes to Post Placement Synthesis (PPS)
- PPS will need to read the new constraint format, the transitions time limits, and the global route for the given placement.
- PPS should do wire sizing was well as gate sizing for two reasons - improving the timing and preventing electromigration problems. If a path is too slow, PPS should look at several options - widening the wire, increasing the spacing from other wires, or sizing the driving gate. By looking at the component and routing density from the global route, it should determine the cheapest of these options and use that.
- From the loads on the net and the transition counts, PPS can figure out the minimum wire width needed to prevent electromigration problems. If it can’t make the wire wide enough (since above a certain width, the capacitance goes up as fast as the allowable current) then it must insert buffers on the net.
- PPS might generate tapered nets by generating pseudoroutes for the detailed router to follow. The global route data will be needed to determine where these pseudoroutes can go without overcongesting the design. It is also possible that widening, or even tapering, the net is not sufficient - in this case PPS must decompose the net into a tree of buffers.
- PPS should enforce slew rate limits in both directions, to avoid signals that transition too slowly or too quickly.
- PPS should insert buffers in lines that are too long. This may not be needed as an explicit operation (the delay, crosstalk, or slew rate rules may apply) but may be useful just to make the delay in long wires as predictable as possible early on.
- PPS should use the global route to determine where to insert any needed buffers. The global route is needed since we will now be inserting far more buffers. Sticking them in where the route already wants to go will result in much less thrashing trying to get the routing to converge.
- PPS must tell the final router about any wire width changes it has made. It could either re-write the global route, put the information in design file, or write a command file for the detailed router.
- Incremental PPS must be used after logic changes. The floorplanner should mark the (relatively few) nets that need to be re-analyzed and re-sized after a logic change.

5.8. Changes to global routing
- The global router must write out the global route in a form a re-synthesis tool can read. This will enable buffer insertion and wire sizing with minimal impact.
- An incremental global routing capability will be needed for use by PPS, so PPS can evaluate different alternatives for improving the design.

5.9. Changes to detailed routing
- The detailed router must follow width, spacing, and tapering restrictions for each net, as suggested by the PPS tool. If any net is much longer after detailed routing than global routing predicted, it may need additional buffering or widening as well.

5.10. Changes to Floorplanning and Flows
- The sub-micron flow must ensure that PPS is run after placement. Otherwise the design may be logically correct, but not work.
- A much more powerful ECO (Engineering Change Order) capability will be needed since most nets will be modified by PPS. The traditional ECO would treat all of these nets as re-defined, and tear them up after any logic change. This must be made much more intelligent, so a logic change can be copied into an already
optimized design without losing the previous optimizations. The ECO program will also need to generate a list of new logic for use by incremental PPS, above.

5.11. Changes to logic verification tools

All the different simulators (Verilog, VHDL, etc.) should be capable of writing out a file that contains the total amount of simulated time, and the number of transitions on each signal during the simulation. This is needed for power estimation and sizing of power busses and signal wires. This file format should be the same for all simulators, and preferably a standard.

There are at least two possibilities for such a format. Power estimation companies such as Sente and System Sciences already use this information - perhaps we could use one of their formats, and encourage them to submit it to OVI for approval. Another possibility is to use existing formats such as VCD (Verilog Change Dump). This has more information than we really need (such as the time when each signal changed) but is already produced by many simulators. It also allows the computation of the currents when averaged over many time scales, which may be needed for some electromigration rules.

6. CONCLUSIONS

There are no fundamental reasons why we cannot build chips with good, reliable deep sub-micron interconnect. However, building tools that follow the necessary guidelines automatically and transparently will require a lot of work. Almost all tools in the chip design flow will need to be changed.

A WORST CASE TIMING AND WORST CASE CROSSTALK ARE NP COMPLETE.

This can be shown by transformation from logic equivalence. Given two logic circuits, it is easy to construct a case where a timing error, or a crosstalk problem, occurs if and only if there is an input vector that causes the two circuits to give different results. Therefore if we could reliably solve the crosstalk problem in polynomial time, we could also solve any logic equivalence problem in polynomial time by converting it to a crosstalk problem, and then solving that. So the crosstalk problem is NP complete; a very similar argument shows that the delay calculation problem is NP complete as well.

B DO WE NEED INDUCTANCE?

The big question about inductance is "Where do the return currents go?" If they go "nearby", then the following back of the envelope calculation shows that inductance is not needed, at least for narrow wires. This is true even though edges get faster as processes shrink, because of the increase in the resistive component of wires. The uncertainty of the return path argues for a methodology rule, such as a ground plane, that would eliminate this uncertainty and make for easier calculations.

Assumptions: w is the width of a wire, in microns. The wire is also a distance w from the neighbors, both top, bottom, and sides. The current in the wire varies with time constant τ, where τ = 100ps + w (50 ps for a 0.5 micron process, 10 ps for a 0.1 micron process). The material is copper at room temperature. Then, using the coax approximation for L and the conductance of copper:

\[
L = \frac{\mu}{2\pi} \ln(3) = 0.219 \text{ nH/mm} \quad R = \frac{1}{\sigma w^2} = \frac{17.5}{w^2} \text{ ohms/mm}
\]

The if the peak current is I, the resistive and inductive voltage drops are:

\[
dV_R = \frac{17.5 \cdot I}{w^2} \text{ volts/mm} \quad dV_L = L \frac{I}{\tau} = \frac{2.19 \cdot I}{w} \text{ volts/mm}
\]

The relative error caused by ignoring inductance is then (note that the I and mm cancel):

\[
\frac{dV_L}{dV_R} = 0.125 \cdot w, \quad (w \text{ in microns})
\]

So at 0.25 micron, we have about a 3 percent error, dropping to 1.2 percent at 0.1 micron. And this is with very fast edge rates, and very good conductors.

Some caveats: wide conductors will have a higher ratio of inductance to resistance. Cryogenic or superconducting interconnect will also make inductance more important.

REFERENCES