Automatic Shape-Based Routing to Achieve Parasitic Constraint Closure in Custom Design

Mark Williams, Co-Founder, Pulsic

Abstract

Each smaller sub-micron process technology brings a new set of physical problems for IC designers. Among the toughest of these problems are meeting electrical parasitic constraints and minimizing signal integrity issues in the interconnect routing while still reaching routing completion, controlling power consumption, staying within the specified die-size and speeding time to market. For digital designs, some of these concerns are addressed by automatic place and route tools. However, for custom IC designs, these issues remain largely unaddressed due to the inadequacy of the automation tools. In addition to custom design tools and flows, there is a need for standardization of data, including design constraints, an effort which is starting to gain momentum at the industry level.

This paper details the increasing problem of achieving parasitic-constraint closure during interconnect routing and how a shape-based routing methodology can help to solve these problems automatically while completing the routing of the design.

Introduction

In electronic circuit design, parasitics are unintended electrical effects, including resistance (R), capacitance (C) and inductance (L), caused by the electrical interaction of the various components and wiring structures of the circuit. Using older process technologies, the parasitic effects of the transistors themselves had the greatest impact on circuit performance. Basically, if a silicon designer did a reasonable job of placement, the vast majority of parasitic and signal-integrity issues were solved already prior to detailed routing. Routing tools could simply concentrate on connecting the circuit according to the design rule checker (DRC) rules while minimizing wire length. However, this approach no longer works today. The widespread adoption of sub-micron technologies has brought about a new set of issues that the designer has to consider, and a new set of challenges for the electronic design automation (EDA) software used in the design process.

Prior to sub-micron design, parasitic constraints (including timing) and signal-integrity issues were considered to be second-order effects, but with today's advanced process geometries, these are now first-order effects. Interconnect delay is the major contributory factor to these issues. The problem is that the increasing number of ever-thinner – and therefore relatively "taller" – wires of sub-micron process technologies has less space between each wire, which causes cross-capacitive coupling among other parasitic effects. Therefore, routing technologies need to consider parasitic constraints and effects of the interconnect continuously during the routing process in order to achieve both routing completion and constraint closure efficiently.

The Parasitic Effects Problem

In electronic circuits there are often signal paths whose specific performance limits the overall performance of the whole circuit. For instance, if a certain path is slower to evaluate than all others in the circuit, it will set the maximum speed at which the circuit can operate. Such paths are called *critical paths*.

One of the critical steps in circuit design is to determine the wiring of the signal paths. The delay (or other performance characteristics) of the signal path is determined both by the components in that path and the wires that join them.

In the past, if a path proved to be too slow to evaluate, designers could change the components so as to increase the signal currents (and thus the speed with which it evaluates). Generally, this approach is no longer adequate, as in leading geometries today, it is the interconnect delay that is the dominant factor and not gate-level delay.

Today, it is often necessary to redesign the wiring itself in order to reduce the parasitics. An automatic routing tool is needed for this task due to its complexity. This tool must be able to recognize which paths need to be routed, and how this routing should be done so as to meet overall circuit performance targets.

Generally, automatic routers are designed to minimize the wire length of every path routed, and so minimize the overall die size of the circuit. This will tend to create dense wiring patterns with wires running for a long distance close to other wires. This was not a great problem in older process technologies, as the wires were both wider and more widely spaced. However, in sub-micron fabrication processes with tall, thin wires, this wiring density can increase the capacitance of the signal significantly, and coupling of signals could occur (known as cross-talk).

When a signal needs to have smaller capacitance in order to meet its performance targets, the automatic router may remove the existing wiring pattern for that signal and try to find another pattern which is better (also called "rip-up and reroute"). However, if the router is not conscious of the capacitance of each wiring segment as it is created, and of the impact this has upon the performance of the finished signal, then the result of rerouting will often be as bad as the original wiring.

It is necessary, then, to create a system by which the router can understand the parasitic issues of the wiring as it is created, and can choose wiring patterns such that the signals meet their performance requirements.

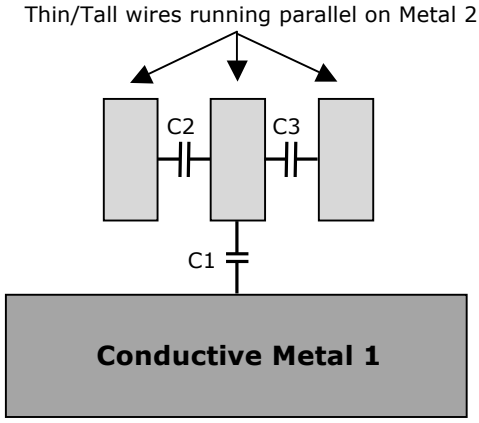

Figure 1: Cross section view of layout wiring. Most of the capacitance of the middle wire is to other wires on the same metal layer.

In Figure 1, the central wire on the second metal layer is part of the critical path signal. Its parasitic capacitance comprises three parts: C1, C2 and C3. The first of these, C1, is the capacitance to the wiring on the next conductor layer below (or to the substrate), the other two capacitances are between other wires on the same wiring layer and the signal itself.

The value of C1 will be lower than either C2 or C3 because the separation of the conductors is larger and, significantly, the facing area is smaller (due to the wires being taller than they are wide). Thus, in dense wiring, most of the parasitic capacitance is to the neighbouring wires on the same layer as the signal.

Figure 2, below, shows a signal being driven by one integrated circuit cell I1 and being received by another cell I2. The critical path involves I1, I2 and the wiring joining them. For a large fraction of the total wire length of the signal connecting the two cells, the wire is adjacent to two other signals on the same metal layer and will therefore have a significant capacitance to both of them. It is seen that the wiring of the critical signal is of minimum length and therefore if it is rerouted it is likely to follow exactly the same path and thus rerouting will not improve the performance of the signal.

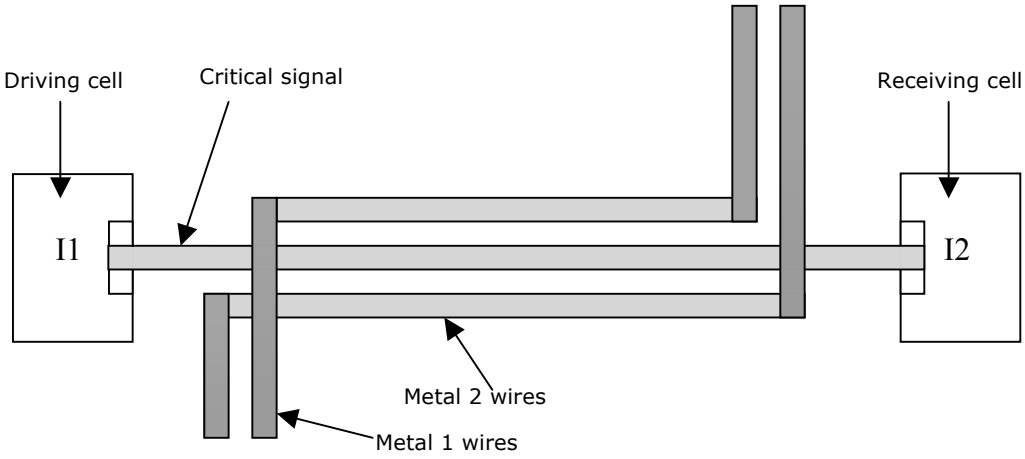

Figure 2: Plan view of integrated circuit wiring

There are two changes that could reduce the parasitics of the critical signal: increase the spacing to the other wires on the same metal layer (as shown in Figure 3); or, find another path for the signal that is of non-minimum length but has lower parasitics than the existing path (shown in Figure 4).

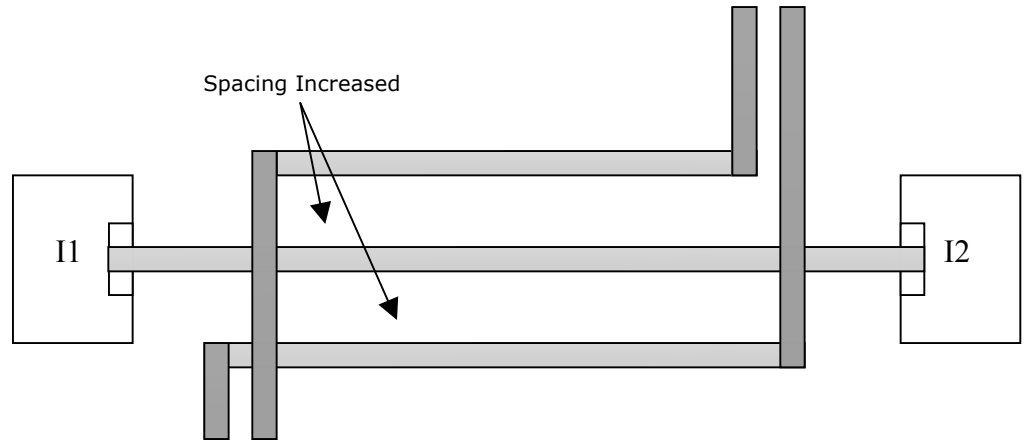

Figure 3: Illustration of the effects of increasing spacing

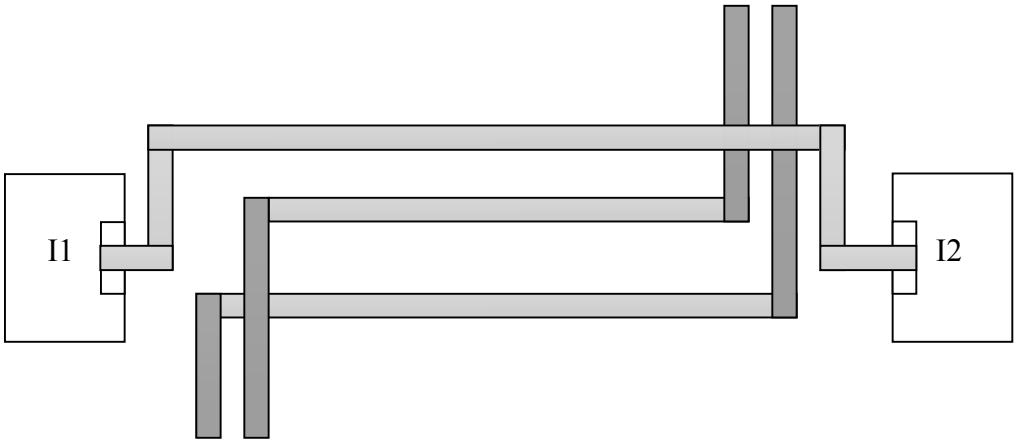

Figure 4: Illustration of the effects of finding alternative path (parasitic constraint driven routing)

Parasitic Constraint Driven Shape-Based Routing

Automatically achieving a routing result such as that shown in Figure 4 requires an intelligent, shape-based routing system closely coupled with dynamic RC extraction so that parasitics can be measured as possible solutions for a route path are found. Unlike grid-based routers, which work on relatively large sections of a design at one time and thus must simplify the design data they consider, shape-based routers only work on the area of the design necessary to route a signal from one point to another, and thus are able to consider full design data and constraints – including parasitic data – during routing.

The automatic routing of a signal, so that it more nearly matches parasitic-dependent constraints, could follow a parasitic-driven routing flow, such as:

(i) analyse constraints down to the net level;

(ii) estimate the distribution of the parasitics along the path BEFORE final routing;

(iii) use this estimate to guide the final routing in an intelligent, cost-driven manner;

(iv) measure the parasitics and consequent compliance with the parasitic constraint after final routing of the net;

(v) apply further steps to improve the routing versus the constraint, and re-measure; and finally,

(vi) if net still fails against the constraint, reroute the net again using this procedure.

Although generally shape-based routing is executed without the need for global routing, there *are* methods for using grid-based global routing techniques to accomplish a parasitic-aware global route. One method for implementation of the first two steps is to perform a global route of the design using a grid of 'bins' per routing layer, then extract the parasitics for each net, using the estimated density of

wires through the routing bins used by that net (and possibly the bins above and below it on neighbouring layers) to calculate the parasitics seen by that net in those bins.

Global Routing Considering Parasitics

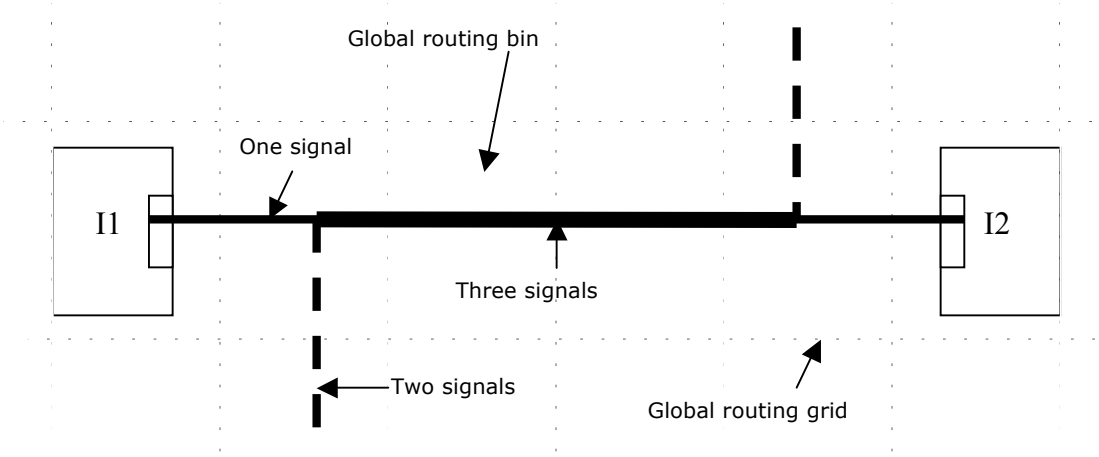

Figure 5: Global routing view of design

Figure 5 shows an example of this kind of global routing. First, the design space is divided up into a grid of bins, either with a regular pitch in X or Y directions, or a variable grid. The rectangles formed by the grid are the "bins." Next the available routing resources for each metal layer in each bin are measured. Then each net is routed through the grid from each source to each sink, using a costing mechanism that penalises using more resource in a bin than was measured as being available. After each net is routed, the resource required for that net is removed from the available amount allotted to each bin by the resource measurement step.

At the end of each pass, any bins that use too much resource are targeted for rerouting in the next pass, during which the nets that use each congested bin are successively rerouted with a higher penalty for using too much resource. Thus, over a sequence of passes, the nets of the design are assigned paths that use no more than the available resource at all parts of the design.

To estimate the parasitics of the nets, it is only necessary to know the center-line path of the net and the expected resource usage along that path. For each bin along the path, the amount of available resource is divided by the number of nets scheduled to route through the bin, and thus the amount of available space per net is found. This space is added to the minimum space needed to route the net legally according to the DRC rules, resulting in the optimal space between each net and its neighbours. This spacing figure, along with estimated densities of wiring on adjacent layers and the distance that the net is expected to run through the bin, enables the early calculation of resistance, capacitance and other parasitic values for the net before the wires are actually put down during the detailed routing phase.

Once all parts of the net have estimated parasitics assigned, then a full constraint estimate is performed. This will give an estimate for how well the net will meet its constraints. As a side-effect, this calculation also yields the contribution of each bin to the total value (i.e., delay) of the net, and also the sensitivity of the overall delay to changes in the parasitic components measured in that bin.

Shape-Based Detailed Routing Considering Parasitics

The same estimates could be derived without global routing, but the information derived needs to be stored in a spatial manner for each net so that the budget and sensitivity values for the relevant parasitics can be derived at any point in space through which the router may be searching for its solution. Other methods for deriving such information might use the minimum spanning tree (MST) or Steiner tree estimations of the paths of the routing.

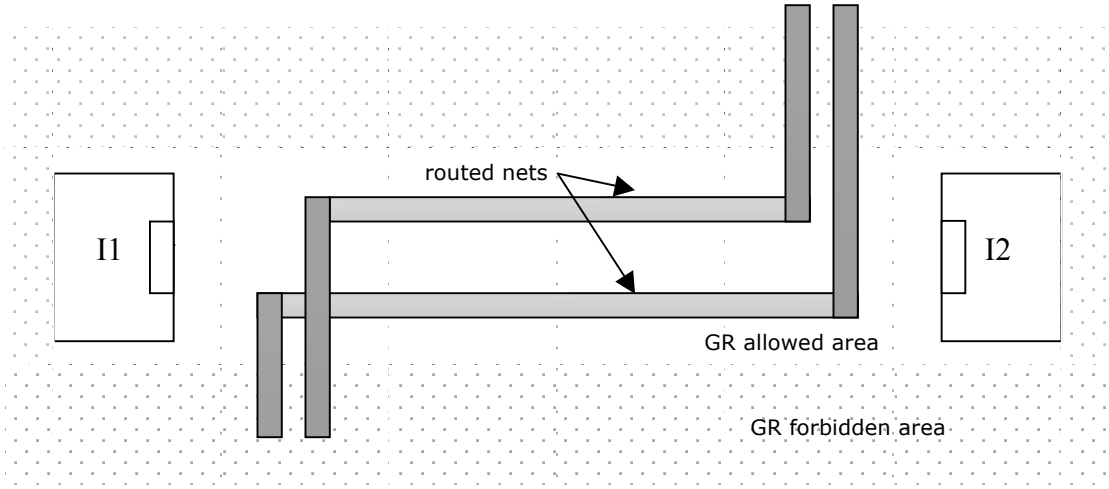

Figure 6: Using global routing result to guide detailed routing

Figure 6, above, shows one method for the implementation of an estimate-guided, shape-based detailed route of the critical net (step iii of parasitic-driven routing flow), using the path determined by the global router to guide the detailed router. This can be done several ways. This example shows that the global routing bins NOT on the global routing path for the critical net can turned into obstacles for the auto-router, thus causing the auto-router to find a solution that is only within the global routing path.

A shape-based detailed-routing methodology aligns exceptionally well with the parasitic-driven routing flow, due to the fact that every exploratory path found during the "flooding phase" (where every free-space rectangle is found proceeding from the source point until the target point is reached, or until no more rectangles can be found) can be checked against accumulated parasitics when coupled with a fast RC(L) extraction engine and cross-referenced against the required constraints to check for errors in real time. Figure 7, below, shows the initial part of the flooding phase of the shape-based, automatic detailed-routing mechanism. A source edge has been formed on the side of the source pin I1 nearest the sink. This has then been expanded toward the right side of the circuit.

There are two components of the costing mechanism that are relevant to constraint-driven routing: the cost of reaching the current point, and the estimated cost of proceeding from that point to a target point. Both of these costs are normally based on spatial and heuristic factors aimed at minimizing wire length, via count and congestion. With constraint-driven routing, however, both of these cost factors can be altered to have additional terms due to the priority to meet the overall constraint.

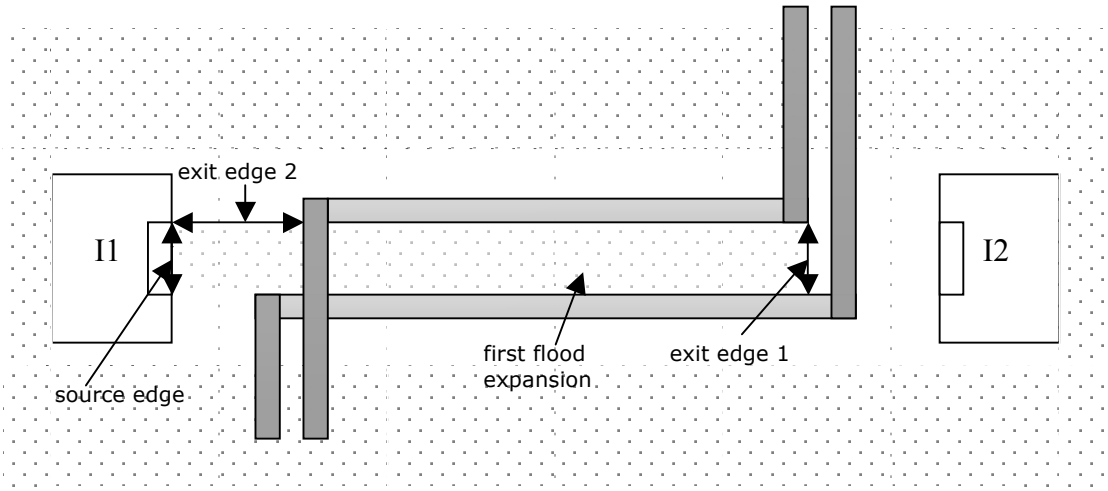

Figure 7: Detailed routing – flooding phase

In the example in Figure 7, the automatic router has found exit edges on the periphery the initial expansion area, which is the area that is legal to route through without a violation against the other signal. Exit edge 1 is normally preferred as it is very close geometrically to the eventual target I2. Normally, exit edge 2 is not preferred both because it is a long way from the target and also involves heading away from the target before getting any closer. Thus in traditional routing, the path would always progress through the sufficient gap between the two other signals and find the target I2 to produce a straight and legal route.

In contrast, at every valid exit edge, shaped-based, constraint-driven routing estimates the parasitics on the path to that edge and checks against the constraints for errors to that point. Exit edge 1 will have a large estimated capacitance due to the narrow flood with close proximity to two other signal wires on the same metal layer. The total constraint error for the path to exit edge 1 is the sum of the errors at the entry edge of the current expansion (in this case the source edge, so nominally zero) plus the errors for each bin within the current flood. The same procedure is used to derive total error cost for exit edge 2 and any other valid exit edges of the current expansion. In this case, exit edge 2 comes from a short path with little or no capacitances, so its accumulated error cost is near zero.

Therefore, in this case, the lowest-cost edge point to expand next can be exit edge 2 rather than exit edge 1 if the accumulated constraint-error cost of exit edge 1 is large enough to outweigh the router's natural target preference. This will lead to subsequent flood expansions as shown below in Figure 8.

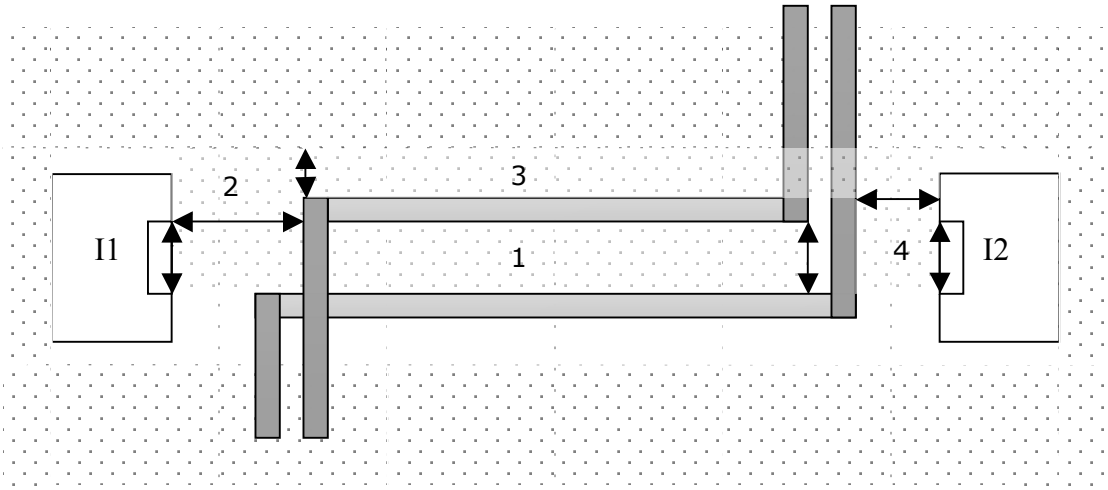

Figure 8: Detailed routing – flooding phase completed

Having reached the sink end of the net with expansions, the router can then choose the best back-track path through the edges (which finds the final position of the wires through the edges) so as to minimize the wiring within those expansions.

Using the mechanism described above, the router can estimate the cost of going through the narrow gap versus going around the far side of the obstacles. The router can choose to take a longer path with lower overall parasitics and thus meet the relevant constraint: even though the length of the path is greater, the interconnect delay is less. The final path through the expansions shown in Figure 8 will as the path shown previously in Figure 4.

Once the net is routed using the mechanism described above, a final check of the net must be performed to ensure that subsequent routing operations have not altered the parasitics.

If the net is failing its constraints due to excessive resistance in a particular part of the net (typically near the source), then the width of the net can be increased ("fattened") in the relevant area, again optionally using the sensitivity values derived earlier to determine the sections of the net where this is relevant.

If the net is failing its constraints due to excessive capacitance in a particular part of the net (typically near the sink), then the net can be moved further away from its near neighbours using route-pushing techniques in the relevant area, again optionally using the sensitivity values derived earlier to determine the sections of the net where such techniques would be effective, as shown in Figure 3.

Both of these are the curative steps (iv) and (v) of the parasitic-driven routing flow. Should the net still fail its constraints due to excessive parasitic, it can be scheduled to be removed ("ripped up") and rerouted in the next pass of the auto-router algorithm, even if the net otherwise meets all design rules and connectivity requirements.

Conclusions

Parasitic effects have a major impact on both routing completion and interconnect delay in today's sub-micron custom designs. Automated, shape-based routing technology, closely coupled with a fast RC(L) extractor and used as part of a parasitic-constraint driven routing methodology, can aid routing completion while managing parasitic effects, minimizing die size and speeding time to market.

As custom design teams use more automation technologies in response to the challenges of sub-micron design, standardization of data formats becomes critical to maintain accuracy and ease the flow from one tool to another. Custom design constraints – such as parasitic constraints – are one area where currently there is no standard format for communicating such data across the design flow. A standard will help more custom designers reap the full benefits of automation technologies such as shape-based routing.

To learn more about shape-based routing technology, parasitic-constraint driven routing methodology and industry efforts to standardize constraints throughout the custom-design cycle, go to http://pulsic.com