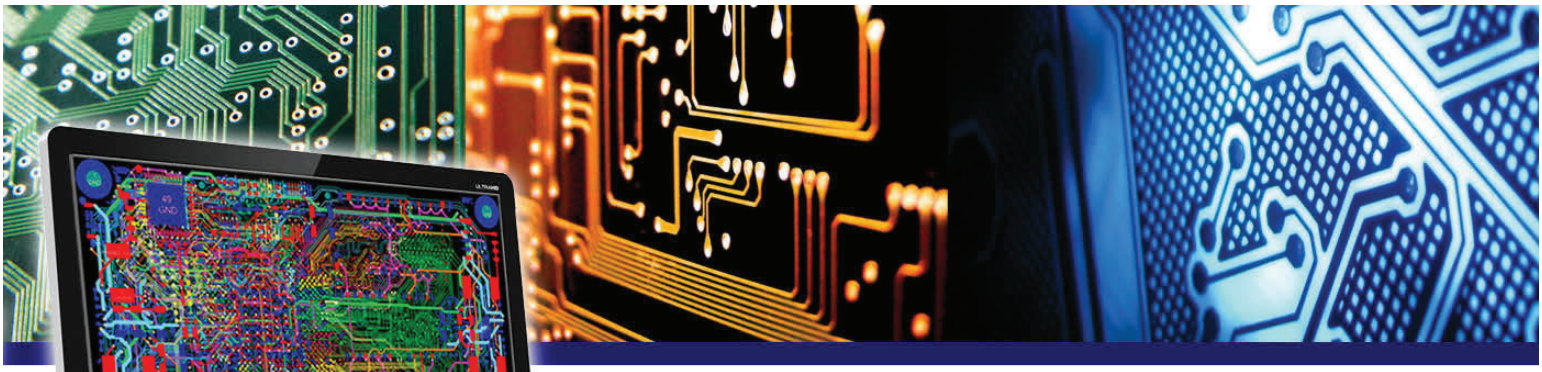

BOARD LEVEL SIMULATION SPECIALISTS

ICD Stackup Planner - offers engineers/PCB designers unprecedented simulation speed, ease of use and accuracy at an affordable price

- 2D (BEM) field solver precision
- Characteristic impedance, edge-coupled & broadside-coupled differential impedance
- Unique field solver computation of multiple differential technologies per stackup
- Heads-up impedance plots of signal and dielectric layers
- User defined dielectric materials library - over 16,250 materials up to 100GHz

ICD PDN Planner - analyze multiple power supplies to maintain low impedance over entire frequency range dramatically improving product performance

- Fast AC impedance analysis with plane resonance
- Definition of plane size/shape, dielectric constant & plane separation for each on-board power supply
- Extraction of plane data from the integrated Stackup Planner
- Definition of voltage regulator, bypass/decoupling capacitors, mounting loop inductance
- Frequency range up to 100GHz
- Extensive Capacitor Library – over 5,250 capacitors derived from SPICE models

iCD
In-Circuit Design Pty Ltd

LEARN MORE

www.icd.com.au

Routing Techniques for Complex Designs

by Barry Olney
IN-CIRCUIT DESIGN PTY LTD AUSTRALIA

SUMMARY: When we analyze customer designs, we often find that crosstalk is a recurrent major issue with manually routed boards. Autorouters are ideal for digital designs as they tend to use all available space, thus reducing the possibility of crosstalk due to proximity. An autorouter is not a push-button solution. However, with a little interactive control and the systematic process outlined in this column, it can be a powerful productivity tool.

To err is human; to completely mess it up, use software.

Autorouter software is essentially artificial intelligence (AI) software – although fairly basic – that makes certain decisions that mimic what designers do in the process of routing a board. Its capacity to do this, of course, varies by software developer, and is dependent upon algorithm complexity and how easy or difficult it is to control the router. Only so many rules can be practically defined, and every situation is different, requiring unique tradeoffs. The limiting factor with any autorouter is describing just what it is that human decision-makers actually do.

Very often, and most especially in tasks that are highly routine and subconsciously automated, designers may struggle to describe all the steps and conditional rules they employ. Not because they do not want to, but because there is simply a lot that they do not think about in an explicable way.

Layout of an analog circuit or a switched-mode supply, and especially one that incorporates specific placement, routing, thermal and isolation requirements – combined with aesthetic goals – relies on many tradeoffs that a seasoned designer would have trouble describing. Many of those tradeoffs involve a series of complex what-if analyses, like multiple possibilities in a game of chess.

Also, complex digital designs incorporating DDR2 or DDR3 memory, for instance, require matched-length routing of all signals, keeping flight times tight between the clock and address signals and the strobe and data signals. Even with all conditional rules defined, this type of routing requires focused interactive control.

It can be a challenge to get PCB designers to use an autorouter because it introduces unknowns: What it is capable of? How to control it? How much time can it save? Instead, many designers prefer to go back to their comfort zone and complete all connections manually using the autorouter between their ears. This internal dialog ends up being a waste of valuable time and can lead to other problems down the track. Some use an autorouter as a sanity check – if the autorouter can route the board to completion, then they can probably do a better job. But there is more to it.

Employed in the proper context, autorouters can make PCB designers a good bit more productive. For example, they can be used to:

ROUTING TECHNIQUES FOR COMPLEX DESIGNS *continues*

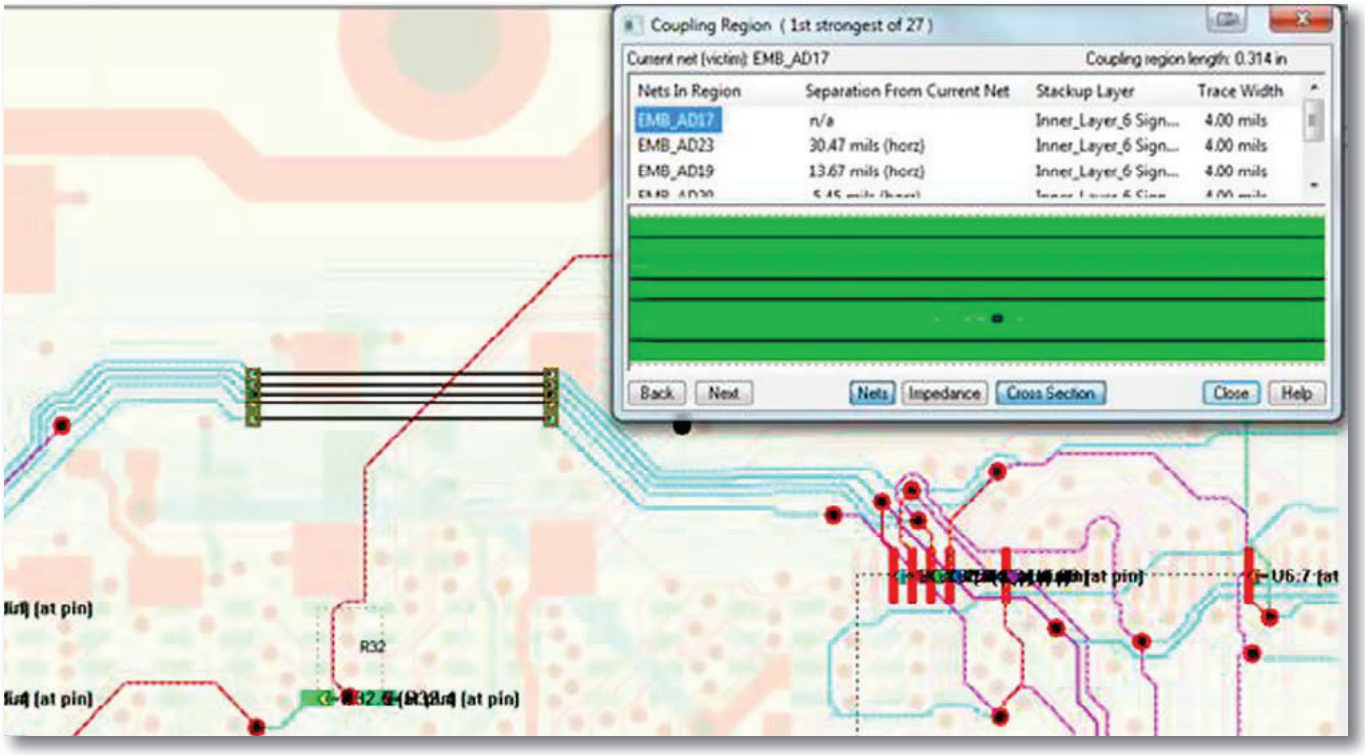

Figure 1: Crosstalk between manually routed parallel segments.

1. Check if design rules have been defined correctly
2. Identify placement congestion
3. Quickly try different routing strategies (after making a backup)
4. Efficiently fan-out from devices
5. Maximize real estate utilization

My company, In-Circuit Design (ICD), provides simulation services such as analyzing customer designs for signal integrity, timing, crosstalk, and EMC issues. More often than not, we find that crosstalk is a recurrent major issue with boards that are manually routed. When we manually route, we tend to use our artistic talents too much, keeping everything nice and neat, coupling traces close together (especially buses) mainly for aesthetics. This may be fine for analogue and low-frequency designs but when we get into the high-speed domain, with rise times < 1 ns, we can only run two trace segments in parallel for less than half an inch before we get excessive crosstalk. Autorouters are ideal for digital design as they tend to use all available space, thus reducing the possibility of crosstalk due to proximity.

Of course, a designer can spend hours setting up design rules to control the autorouter, but I prefer to drive the autorouter from the schematic. This only requires the setup of the most basic rules. When we draw a schematic, we draw it by functionality, and I believe that we should also place and route by functionality. In this way, I can add my own creativity and decision-making on the fly, while still taking advantage of the automation.

Most popular EDA tools have the ability to cross-probe between the schematic and router. This is a fantastic feature that enables a PCB designer to build up an extremely dense, complex route, in a couple of hours – by controlling the router from the schematic. I discussed this in detail in my previous Beyond Design column: Interactive Placement and Routing Strategies.

We do not need to do any routing ourselves to get an acceptable route of the non-critical nets. Of course, matched lengths, differential pairs and other critical signals should be routed with the precision they require. I start by placing all the components by functionality, selecting the desired component on the schematic and dropping them where I want them on the

Figure 2: Cross-probing from schematic to router of part of a PCI bus.

PCB. Similarly, when routing, I select a chip on the schematic, the nets are highlighted on the PCB, and I select "Fan-out" on the router. Then, select the critical nets on the schematic "Fan-out," and "Route" with the autorouter. I use the "Move" command to push and shove the traces where I want them, then move on to the next group of nets and repeat. Each group of routed traces is then verified after completion.

When we drive the router from the schematic, it's possible to see what needs to be done without entering conditional design rules, and we can later manipulate the traces as if we hand-routed them. Once all the critical nets are routed, I fix them, then turn the autorouter loose on the remainder of the nets to finish off the connections.

The Perimeter Routing Technique

I have used the following technique successfully, over the years, with a number of autorouters. It was first put to the test on the Daisy/Dazix Star Router back in 1987, Cadence Prance-XL Router, Mentor's Expedition Autoactive, and then the PADS Router.

In other words, it's safe to say that the technique generalizes.

Let's assume that you have followed a methodology for placing and routing critical, high-speed signals, fixed them, and that the remainder of the signals are non-critical. We will look at a scenario in which the board is 98% completed after a series of autorouter and manufacturing routines. A 98% completion rate sounds pretty good, but we all know that the router will leave the most difficult/longest traces for us to complete. Indeed, the last 50 or so traces may take us days of head-banging frustration to complete.

If a board will not route to completion, it may not be the router's fault. It could just be that we have: (a) poor placement with bus crossovers, (b) poorly defined design rules, or (c) have not planned enough signal layers into the stackup. I guess you get a feel for how many layers are required after doing a few boards. My general rule of thumb is that if I cannot get at least 85% completion before I start tweaking the design, then I will have serious problems. With less than 85% completion out of the blocks, I re-evaluate component placement, redefine design rules if necessary, add a couple more signal layers, or reduce the functionality of the design.

All routers tend to route inward because the algorithms are tuned to make the shortest possible connection of the two open ends. This is why you generally see a tangle of rats' nests in the centre of the board where all the signals try to cross. Beneath this apparent obstacle is an underlying opportunity.

Here's the trick: Define a route keep-out perimeter channel 200 mils around the edge of the board. Avoid enclosing component pins that have connections, as they also need to be

ROUTING TECHNIQUES FOR COMPLEX DESIGNS *continues*

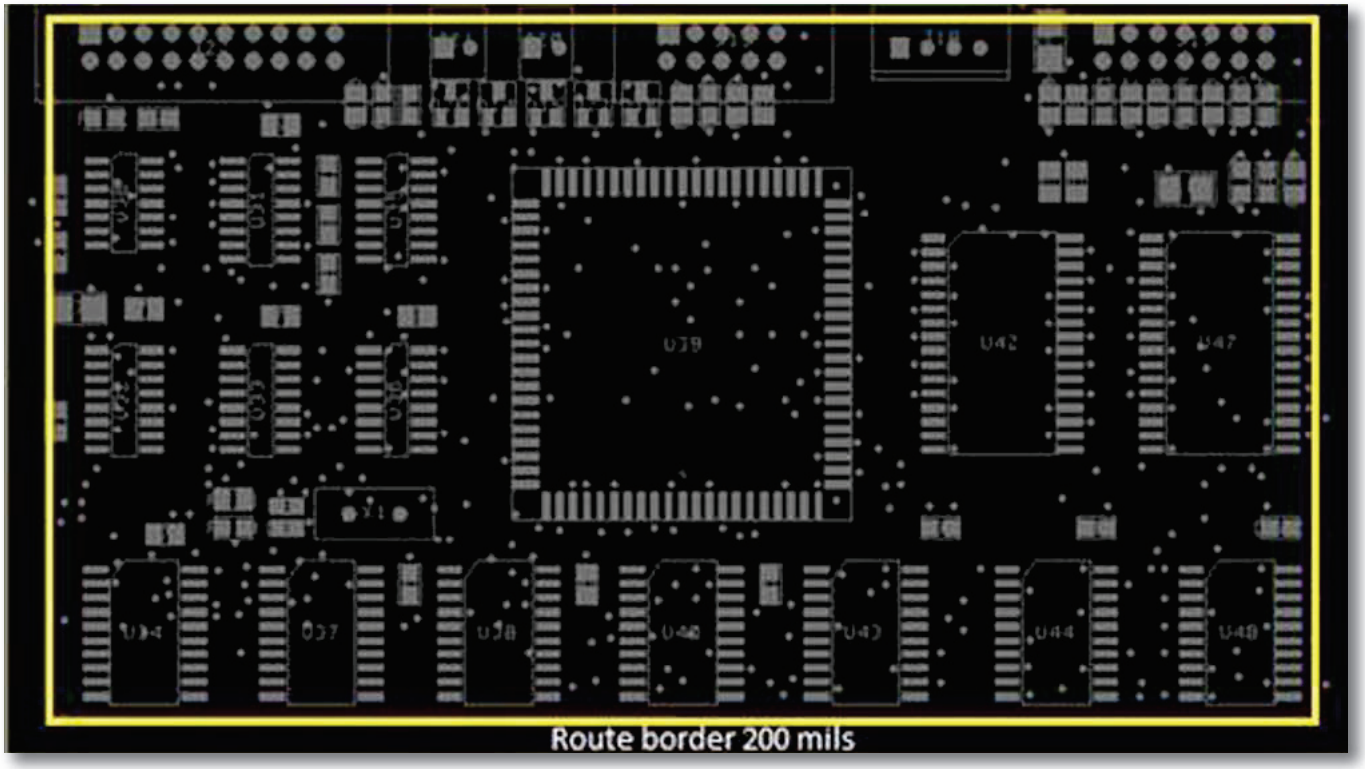

Figure 3: Route border defined 200 mils for the edge of board.

routed. Most components should be 200 mils from the edge anyway, if you are following IPC standards, but there will be connectors and interface devices, etc., that are closer. This channel will prove to be invaluable later to polish off that last 2% of nets.

Autoroute the board to the best completion rate. Some rip-up and retry of connections should be tried and the via minimization and manufacturing passes should be utilized. The autorouter will smooth the lines, remove ghost vias and staircases, eliminate unnecessary vias and reduce the etch length. But, we still have nets to route.

A via fan-out grid should be used initially to avoid blocking route channels. However, at this stage the via grid can be removed as we are only interested in completing remaining connections, since the main routes are in place. Invoke the autorouter again.

Finally, drop the route border to 50 mil perimeter. This gives us 150 mils of extra routing channel on all signal layers around the perimeter of the board. Using 5/5 technology this equates to 15 additional traces per layer. To complete the remaining nets, manually route each net out to the edge, follow the perimeter around the board in either a clockwise or anticlockwise direction, and then route back in to terminate the connection. This provides an additional 30 traces per signal layer; for example, on a 12-layer board with 8 signal layers, that's 240 additional traces, typically more than enough to complete the route.

Points to Remember

- Autorouters are tools that can make designers more productive, but they don't represent a trivial, push-button solution.
- The limiting factor is describing just what it is that human decision-makers actually do. Many of those judgments involve a nested series of complex what-if tradeoffs.
- Autorouters can be particularly useful for digital designs, as they tend to utilize all available board real estate, reducing the possibility of crosstalk due to proximity.
- A designer can spend hours setting up

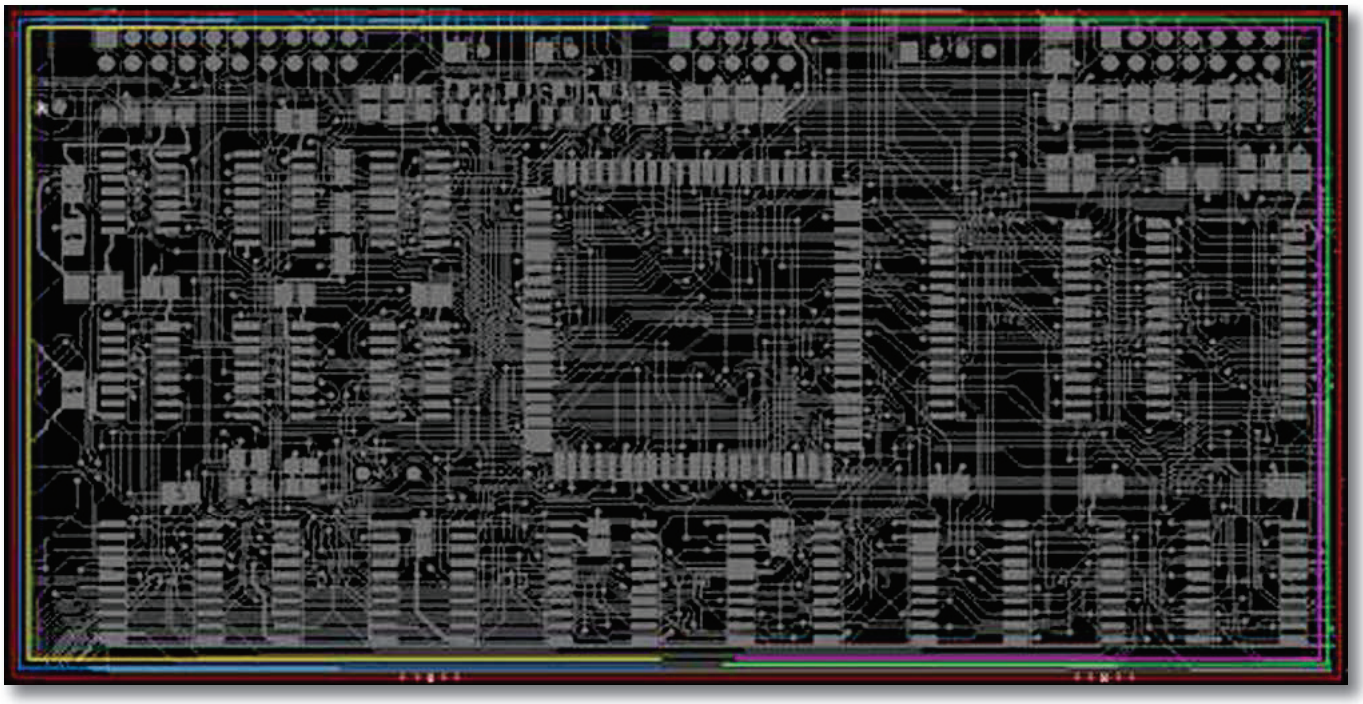

Figure 4: Route border reduced to 50 mils from the edge.

design rules to control the autorouter. But, when we drive the router from the schematic, it's possible to see what needs to be done without laboring through the process of dialing in complex tradeoffs and conditional design rules.

- Cross-probing between the schematic and router is a fantastic feature that allows the designer to build up an extremely dense, complex route in a couple of hours.
- As general rule of thumb, if you can't reach at least 85% completion with the autorouter without manually tweaking the design, you're setting yourself up for some serious design headaches, as you finish the board.
- If a board won't route to completion, it may not be the router's fault. It may just be that we have: poor placement with bus crossovers, incorrectly defined design rules, or we haven't allowed enough signal layers in the stackup.
- The perimeter routing technique provides an additional 30 traces per signal layer. For a 12-layer board with 8 signal layers, that's a whopping 240 additional traces.

PCBDESIGN

References

1. Advanced Design for SMT – Barry Olney
2. Beyond Design: Interactive Placement and Routing Strategies – Barry Olney
3. Beyond Design: Intro to Board-Level Simulation and the PCB Design Process – Barry Olney
4. Beyond Design: Mixed Digital-Analog Technologies – Barry Olney
5. PCB Design Techniques for DDR, DDR2 & DDR3, Part 2 – Barry Olney
6. PCB Design Techniques for DDR, DDR2 & DDR3, Part 1 – Barry Olney
7. The ICD Stackup and PDN Planner can be downloaded from www.icd.com.au

Barry Olney is managing director of In-Circuit Design Pty Ltd (ICD), Australia. ICD is a PCB design service bureau specializing in board-level simulation. The company developed the ICD Stackup Planner and ICD PDN Planner software.