

**Figure 3:** Workflow diagram circuit and layout interaction

The layout planner is a very important component of the circuit design environment. It has become necessary to incorporate accurate layout information (e.g., interconnect delays) during the circuit design stage in order to reduce design iterations. These iterations are necessary when the final layout does not satisfy all the assumptions made during the circuit design phase. Generally, circuit and layout design are done by different designers, and contemporary tools inadequately capture the design assumptions in the existing file formats. Layout planning of blocks is used to obtain early estimates for block area and timing of critical signals. The layout-based estimates are used during the circuit design stage to carry out more accurate circuit simulations and to design the datapath circuit schematics. The same layout plan is later used to drive the layout synthesis process. The layout planning methodology is designed with the following goals:

- be fast and highly interactive
- provide reasonable estimates for area and interconnects
- drive layout synthesis with a place and route plan
- enable what-if analysis and provide tradeoffs between accuracy and tool performance

The layout planning flow is developed from experience with prototypes used in some recent microprocessor design projects. Early layout planning provides a user with a means to estimate the layout area of a datapath block and the associated interconnect parasitics, which are used to perform quick performance verification analysis. The information obtained is then used to complete the design. The layout planning flow provides various trade-off points so that the circuit designer can get better estimates on interconnect design at the cost of tool performance. The user

can estimate interconnect parasitics derived from minimum spanning trees for rough estimates and actual global routes for more accurate estimates.

### Functionality

The inputs to the layout planning tool are design constraints, user-defined placement hints, and the netlist (which may be incomplete). The tool provides a means to visually see and edit the placement and change the netlist. The netlist is modified if a cell used in the block is changed because of higher drive strength requirements, or any other interconnect optimization need. The key functions performed in the layout planning stage for interconnect optimization are as follows:

- cell area estimation and interface design
- placement of cells (as part of vectors)
- identification of vectors and rows
- global routing and congestion analysis
- parasitic estimation and timing analysis

### Placement Modeling

A key feature of the datapath layout planning is the type of layout modeling that is used. Due to multiple instantiations of logic cells, common in datapath blocks, layout editing provides a means to edit groups of instances in one command. The multiple instances of a cell are grouped into entities known as vectors, the contents of a stage in circuit design are placed in a row, and the contents of a bit-slice form a column. Thus, the complete layout plan is modeled as a matrix. The tool then provides commands to move, delete, create vectors, rows, matrices, etc. This kind of modeling aids in ensuring regularity in the placement of cells in the layout plan.

### Interconnect Estimation and Optimization

During layout planning the design engineers need to estimate the interconnect delays so that better data can be input to the circuit simulation stage. The interconnect length can be estimated by generating the Steiner tree [1, 2]. This estimation generates optimistic net lengths, as trees are generated for one net at time, and no consideration is given to obstructions or congestion due to other nets. For this reason, net length estimation also has a quick runtime.

Better net length estimates are generated by doing global routing. Global routing accounts for obstructions as well as congestion. It also considers physical net specifications (width, spacing). This option for net length estimation is slower than the Steiner estimation.

For clock and other critical nets, some other tree estimation algorithm such as A-Tree [3] may also be used.

The layout planning tool provides these choices as runtime configurable user options.

### Track Share Analysis

After a reasonable placement has been determined, the user is able to estimate interconnect parasitics. The location of the interface ports of the cell can also be planned to enable better routing [4, 5]. A typical routing, shown in Figure 4a can be improved with cell interface ports planning as shown in Figure 4b. The interface planning can be carried out using Track Share Analysis (TSA) or global routing. Based on the results of global routing or TSA, the interface terminals of the cells are placed at appropriate locations. The net length estimation process was successfully used in a recent project. By providing additional planning capabilities, we have given the designer full control of top-down as well as bottom-up aspects of datapath block layout design.

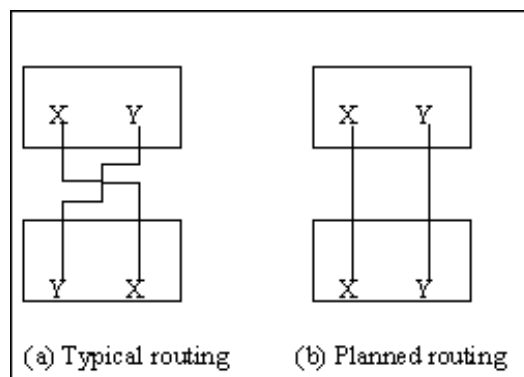


Figure 4: Cell interface planning

### Visualization

The interactive graphical user environment provides other features. The user can plan for routing space and analyze routing congestion information that is derived from global routing. Based on the congestion analysis, the user can manually adjust the placement and plan out for area. The tool provides net visualization and editing functionality to interactively optimize the interconnect delay.

FCDE provides path viewing and debugging capabilities and includes the following viewers:

1. Paths list viewer. This is a list of all paths, including path properties (start point, end-point, margin, and more). The path list has links to a detailed path viewer, a schematic editor, and a layout planner.

2. Path detailed viewer/tracer. This viewer serves as the main debugging aid for speed-path analysis and optimization.

### **Incremental Design**

The layout planning stage involves a lot of what-if analysis work. For example, the user may want to change the placement of a few instances and see how the area or timing on those affected nets has changed. To aid the user, the layout planner provides for incremental design. In this mode, the tool detects what nets have changed, and re-computes the desired properties for the affected nets only. The tool is also able to display the delta changes (difference between the property's old value and the new value). This allows the user to see immediately if the changes made are having positive or negative effects. Another advantage of incremental design is the efficient runtime, which is very important for an interactive design tool.

### **Cell Area Estimation**

As layout planning happens before any real layout is created, an important requirement of the layout planner is to be able to estimate cell area from the netlist of the cell. The cell area estimator can provide various choices for estimation. We have estimated cell area in two ways and are experimenting with others.

The first method is to use a statistically derived equation for estimating cell area. The function's parameters include the number of devices, number of p-devices, number of n-devices, and the number of I/O ports. The second method is to use historical data. In this method, a table is created for various common types of cells, and the area of a cell is obtained by matching it against an entry in the table.

Other methods that we are experimenting with include the modification of the core engine of a cell synthesis tool. This method is expected to provide the most accurate estimates, but it is also expected to have the longest runtime.

### **Parasitic Estimation**

Interconnect parasitic (resistance and capacitance) estimation is required since this information translates directly into delay information that can be used during circuit simulation. The parasitic estimation tool is designed to provide various run-time configurable options. These options allow the user to make tradeoffs between runtime and accuracy of the estimates.

### **Timing Analysis**

In order to enable interactive design, a quick timing analysis engine is integrated into the layout planner. The quick engine yields rough timing analysis, but provides reasonable information on changes that occur when the layout plan is modified. Since the timing analysis data (slopes, timing widows, etc.) are shared between many circuit design tools (e.g., driver sizing), the timing analysis tool is a separate component of the circuit design environment. The layout planner invokes this engine when required, and the process of transferring data is transparent to the user.

### **Noise Analysis**

Noise on interconnect is becoming a more visible problem with deep sub-micron designs. At present, the layout planner provides a means to visualize the noise as aggressor-victim pairs. This helps the user plan appropriate spacing between the aggressor and the victim. Current experimental work is directed towards auto identification of aggressors and victims. Based on the net topology, timing vectors on adjoining nets, and estimated cross-coupling capacitors, the proposed tool will be able to compute if there are any signal integrity issues, and then designers can either manually, or with the help of an automated router, reduce the net cross-coupling.

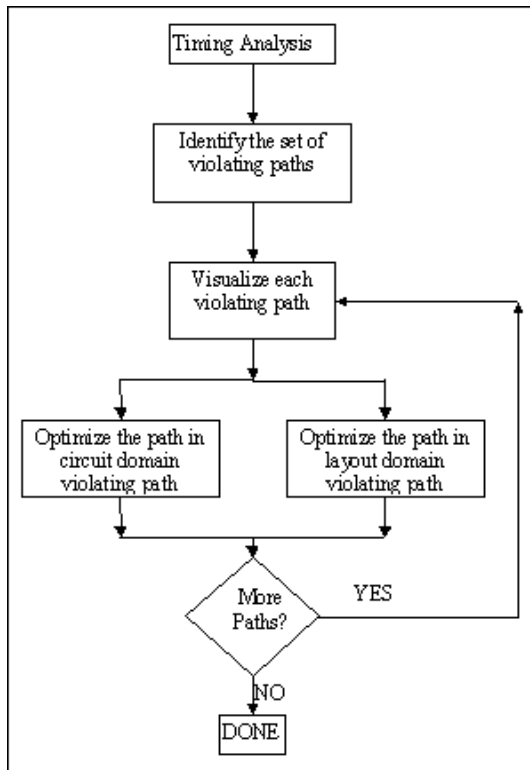
### **Device Size Tuning**

FCDE will provide tight bi-directional communication with the schematic editor, a change notification system, and incremental capabilities. Using these capabilities, it will enable circuit designers to change the size of a device or a cell in the schematic editor or in one of the FCDE viewers. The change will be applied on the FCDE data model, and incremental analysis will be made to analyze this change.

### **New Functionality Enabled**

#### **Speed-Path Design**

We describe the design environment and the advantages of using it by looking at the speed-path design activity commonly performed in IC design. This design activity encompasses both circuit design and layout design. It uses several tools and various types of data in both of the design domains. The flow of this activity is shown in Figure 5.



**Figure 5:** Speed path optimization flow

Speed-path analysis and optimization flow start with the timing verification stage when a circuit designer receives timing constraints and budgeting. The designer runs tools to identify critical paths. To optimize the paths with violations, there are multiple possibilities:

1. Increase the driver strength.
2. Reduce the driver load.
3. Optimize the interconnect loading.

The first two options are handled by the device-sizing functionality in circuit design, and the last one is carried out in layout design.

With current tools, all these activities (shown in Figure 5) are carried out by different tools. The data is exchanged by means of files and often there are multiple design engineers involved. Since data files are used, it is difficult to exchange partial data, i.e., complete design data is exchanged between the tools. By some estimates, it takes in the order of weeks to optimize a path if accurate interconnect loading is to be obtained. The reason for this is that layout design is carried out by a different person and the turn-around

time for obtaining data is long. The most common reasons for the long turn-around time are that layout design has to be completed before data is obtained, and data interfacing is difficult. Thus, circuit designers tend to optimize the paths using device sizing and do not explore all possible solutions, such as optimizing the interconnect delay.

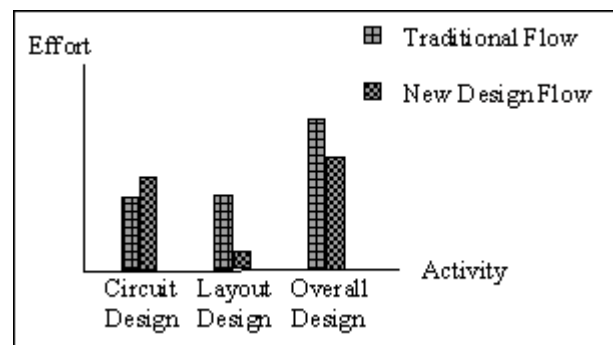
With the new integrated design environment, where all the different tools are accessible via a common user interface, the interconnects can be optimized as easily as devices can be sized. Also, since all the tools are working with the same data model, the data is exchanged in memory. This enables interactive design, which provides an improved turn-around time between various tools. Another advantage of the integrated design environment is that it provides the capability for incremental design. Only data that is modified by one tool needs to be addressed by other affected tools. With the integrated layout planner, a circuit designer is able to make changes to the block layout plan without involving a different person to do the layout design. Changes in interconnect parasitic values are updated in the common data model, and the timing analyzer is able to perform incremental analysis of the change.

### Noise Handling

Another activity the new environment enables is the efficient handling of noise. It has become important to account for noise as the operating voltage for deep sub-micron design is decreasing, and the noise effect is becoming more visible. Noise analysis also involves information that is traditionally spread across both the circuit design and layout design stages. As both of these stages are tightly integrated in the new design flow, all the information required for noise analysis is available simultaneously.

Some of the noise analysis features made available have been adapted from published work [6].

### Results



**Figure 6:** Gains in design time

Our results are illustrated in the bar chart shown in Figure 6. With the traditional approach, it is observed that the effort spent in circuit and layout design was almost equal. It is clear that when layout planning data are available during the circuit design stage, the circuit design time increases (about 1.25X). However, there is a significant decrease in the layout design time, which includes the time spent in fixing layout (about 1/3X). This reduces the overall design time by about 25%.

For a typical datapath block, the average time for post-layout fixing of critical paths improved from over two weeks to less than a week when the prototype of the proposed integrated design environment was used.

### Future Work

Based on the layout plan, top-down cell templates are generated that need to be synthesized. During circuit design, these cells are placed and routed as per the interconnect plans. However, if a cell cannot be synthesized using top-down planning, a bottom-up correction to the overall block plan is generated. If several such iterations occur, a significant productivity penalty will be incurred. New cell synthesis algorithms are needed that can handle the top-down topology plans. Another area of exploration is to combine layout information even earlier in the design cycle, namely during logic synthesis. This will enable a designer to appropriately choose static or domino technology to meet the timing or area constraints and also insert the right number of pipeline stages during datapath design. All these call for a modification of traditional logic synthesis algorithms to account for layout effects.

### Conclusion

By integrating circuit design with layout planning, we have improved the overall design time. Using early layout planning, we are able to incorporate accurate interconnect parameters into circuit design. This generates better timing analysis results, which match the actual post-layout timing analysis, thus reducing the need for re-design and re-layout.

Results from a recent microprocessor design project support the need for layout planning by showing that the amount of re-design and re-work required is reduced for blocks when early layout planning is carried out. In the speed path flow, expert design engineers have observed a three to five times improvement rate in the time it takes to fix violating paths.

### Acknowledgments

We especially acknowledge Naresh Sehgal for valuable discussion and comments on the paper. We thank

Ehud Kedar (former FCDE group leader), Gil Amid (Nike Timing analysis and circuit simulation group leader), Eitan Zahavi (Nike system architect), and Paul Madland (Intel Fellow, PMD circuit technology group) who have all contributed significantly to FCDE architecture and vision. We also thank Anurag Gupta, V Nagbhusan, and Manoj Gunwani from the Santa Clara NIKE department, and Anat Ben-Artzi, Ronan Moldovan, and Arkady Neyshtadt from the Haifa NIKE department. We also acknowledge various members of other NIKE groups and the design teams who contributed to the realization of this methodology.

### References

- [1] M. R. Garry and D.S. Johnson, "The Rectilinear Steiner Tree Problem is NP-complete," *SIAM J. Appl. Math.*, Vol. 32, No. 4, pp. 826-834, 1977.
- [2] J.P. Cohoon, D.S. Richards, and J.S. Salowe, "A Linear-time Steiner tree routing Algorithm for Terminals on the Boundary of a Rectangle," *Digest of Technical Papers, ICCAD-88*, pp. 402-405, Nov. 1988.
- [3] J.J. Cong, K.S. Leung, and D. Zhou, "Performance-driven interconnect design based on distributed RC delay Model," *Proceedings ACM/IEEE Design Automation Conference*, 1993, pp. 606-611.
- [4] B. Krishna, C.Y.R. Chen, and N. Sehgal, "Technique for Planning of Terminal Locations of Leaf Cells in Cell-Based Design," *Proceedings of 11<sup>th</sup> International Conference On VLSI Design*, pp. 53-58, 1998.
- [5] Amnon Baron Cohen and Michael Shechory, "Track Assignment in the Pathway Datapath Layout Assembler," *Digest of Technical Papers 1991 IEEE International Conference on Computer-Aided Design*, pp. 102-105, 1991.
- [6] D.A. Kirkpatrick and A.L. Sangiovanni-Vencentelli, "Techniques for Crosstalk Avoidance in Physical Design of High Performance Digital Systems," *Digest of Technical Papers, ICCAD-94*, pp. 616-619, November 1994.

### Authors' Biographies

Bharat Krishna is the layout planner project leader in the NIKE/DT department. He received a M.S. degree in computer engineering from Syracuse University in 1994 and a B.Sc. degree in electrical engineer-

ing from the University of Khartoum, Sudan in 1991. He has worked for Intel since 1995 in the datapath layout automation area. His interests include datapath layout automation, VLSI routing, and physical CAD tool design. His e-mail is [bharat.krishna@intel.com](mailto:bharat.krishna@intel.com).

Gil Kleinfeld is the FCDE group leader in the Nike/DT department. He received a B.Sc. degree from Tel-Aviv University in mathematics and computer science. Gil has been working for Intel since 1988 in the areas of logic synthesis and datapath automation. Gil's main interests are automation of tedious design and verification tasks, and software design. His e-mail is [gil.kleinfeld@intel.com](mailto:gil.kleinfeld@intel.com).