# A Pseudo-Hierarchical Methodology for High Performance Microprocessor Design

A. Bertolet, K. Carpenter, K.Carrig, A. Chu, A. Dean, F. Ferraiolo, S.Kenyon, D. Phan, J. Petrovick, G. Rodgers, D. Willmott
T.Bairley✞, T. Decker✞,V.Girardi✞, Y. Lapid✞, M. Murphy✞, P. A. Scott✞, R.Weiss✞
IBM Corporation, Essex Junction, Vermont
✞Cadence Design Systems, San Jose, California

***Abstract* - This paper reports on a highly effective methodology to construct complex high performance microprocessors. Critical aspects of the methodology include an integrated database for design control, algorithmic power grid generation, fully customized clock network insertion, timing driven placement and routing, an integrated timing closure strategy, and incremental checking that includes formal netlist verification, DRC and LVS. The methodology places particular emphasis on continuously improving the integration process and incrementally improving both the design and the interoperability of the tools. The final chip tape-out was 17 calendar days from the final netlist.**

## I. INTRODUCTION

The current generation of high performance superscaler microprocessors pushes chip transistor counts of 3-5 million. The architectural complexity and cycle time requirements create critical challenges in clock distribution, robust power distribution, timing driven design, engineering change management, and full-chip verification. To deal with these challenges, a design methodology was crafted to rapidly and repeatedly integrate the chip while providing fully customizable clock and power distribution and an automated engineering change (EC) facility to deal with last minute timing and logic corrections. A "pseudo-hierarchical" design approach was applied to manage the floorplan, placement, and routing optimization of the standard cell control logic. The methodology is an evolution from a traditional hierarchical approach and results in a die size reduction of 96mm$^2$ on the same netlist.

The methodology was applied to a highly superscaler PowerPC™ microprocessor. The chip was fabricated in an 0.35µm ( $L_{eff}$ = 0.25µm), 2.5V CMOS n-well technology with six levels of metal: five for global routing and one for local interconnect. The die, as depicted in Figure 1, is 10.4mm X 14.4mm and contains 6.5 million transistors. The chip includes 48KByte of cache and 121 custom macrocells. Random control logic is implemented with approximately 67,000 standard cells instantiated at the top level of the hierarchy. In addition, there are approximately 32,000 master-slave latches (25,300 of which are in custom macrocells) that require a highly tuned clock distribution network.

The design environment consists of third-party and proprietary IBM CAD tools interfaced to Cadence Design Framework II™ (DFII). DFII is used for data and design process management. Using the DFII extension language, SKILL™, enables the team to customize and extend the environment as needed.

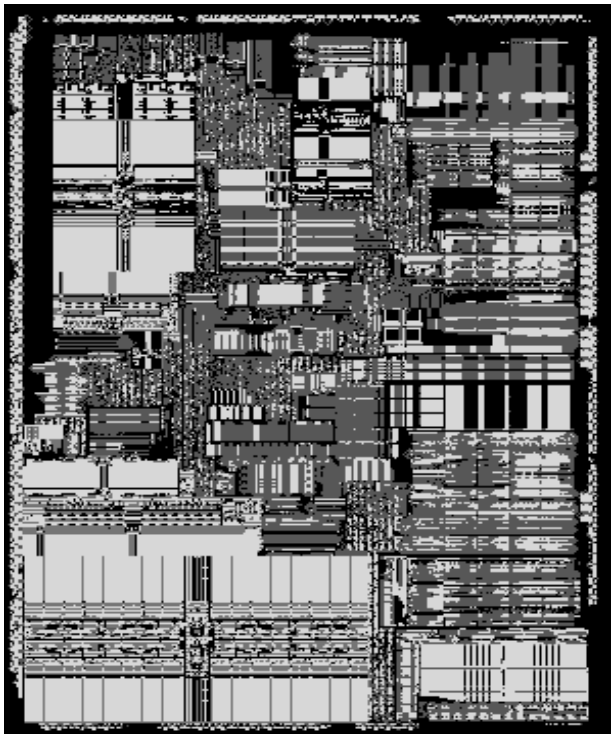

Figure 1. Chip layout (metal layers not shown)

## II. METHODOLOGY OVERVIEW

The cornerstone of the design methodology is the ability to rapidly and repeatedly integrate and verify the chip. The notion of "construct by correction" was utilized to repeatedly integrate the chip over a three month period. Each iteration incrementally improved the overall quality of the design, and enabled real-time feedback of parasitic RC values to the timing tool. In order to iterate on the design, a consistent design view of VHDL, synthesized netlist, and physical design data is essential. As such, facilities were developed to enforce consistency by creating incomplete or missing physical design data such that it matched exactly the specifications of the incoming netlist (Figure 2). This enabled floorplanning, placement, and routing to continue, despite errors, throughout the earlier phases of the integration

process to provide valuable timing feedback with routing parasitics. Any problems encountered are corrected concurrently and captured on the next pass through the process. In addition, techniques were developed to make early full-chip DRC and LVS assessments without final routing data. The net result of the incremental process was a final netlist to tapeout cycle of 17 days, which included two EC passes, a crosstalk assessment, and a fast-path assessment.

Despite the fact that the focus of the methodology is on iterative improvement and rapid integration, technical capabilities are not sacrificed. The final integration cycle includes a fully custom, algorithmically generated power distribution on five layers of metal that includes legal standard cell core site definition based on power analysis. In addition, a fully customized clock distribution network is generated with a resultant typical skew of 180pS. Furthermore, multiple routing passes are performed to handle special classes of nets. The methodology focuses on tight integration of these sophisticated capabilities to maintain quality and productivity.

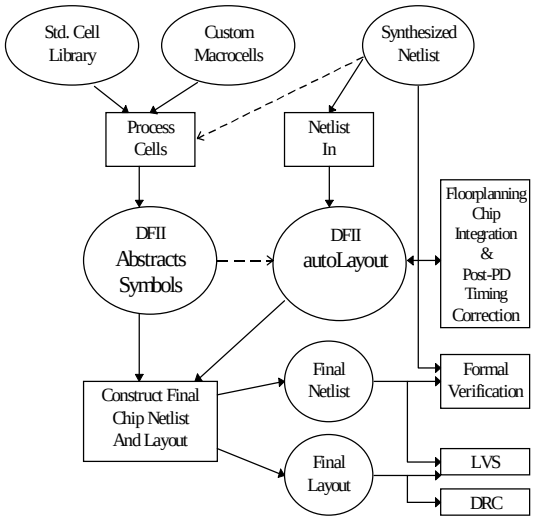


Figure 2 Methodology Overview

## III. DATABASE AND METHODOLOGY CHECKS

The chip integration methodology discussed above uses the DFII database to represent all chip design views from the original synthesized netlist to final chip. The DFII database models both the logical and physical structure of the chip so all changes could be represented.

There are two key components to the database:

1. The DFII autoLayout view contains full-chip netlist, placement and global routing information.
2. The DFII abstract views for macrocell and standard cell components (essential placement and routing data derived from the layout views).

Critical to the "construct by correction" method is the ability to rapidly integrate incrementally improved versions of the design. A facility called "Process Cells" manages and enforces design consistency. The full-chip synthesized netlist is the "golden" chip representation. Process Cells analyzes the netlist and checks for a unique symbol in the standard cell or macrocell library referenced by each instance in the netlist. If a symbol is missing, or has incorrect terminals, a symbol is created which matches the terminals required in the netlist. The symbols are an important link to the original netlist, since they will be referenced to create a full-chip schematic netlist for final verification. Next, an inventory of layouts that corresponds to the symbols is built. If a layout is missing, a "black box" consisting of a place and route boundary can be generated automatically or by the macrocell designer. Finally, an abstract is created from each layout. During abstract generation, the symbol and layout are examined for consistency. Terminals in the layout but not in the symbol are deleted, while terminals in the symbol but missing from the layout are created. At this point, Process Cells has ensured that a consistent set of symbols and abstracts are available to construct a full-chip autoLayout view of the synthesized netlist, and to proceed with floorplanning, placement, and routing. This innovative approach allows floorplanning, placement, and routing to occur with *no* layout, symbol, or schematic cellviews, and to incrementally add them to the chip integration process as they are completed.

In order to avoid problems in the abstract generation process, macrocell designers are provided with a methodology checking program. This program ensures that layouts meet guidelines established for placement and routing. The key checks are:

- Pin names match names found in the netlist.
- Pins are on-grid.
- Pins are on correct layer and accessible to router.
- Manufacturable shapes are enclosed by a place and route boundary by half the minimum design rule space.
- Clock pin(s) meets the clock router methodology.

Process Cells provides an important link to the macrocell design team. The system is highly automated and is designed to tolerate problems in the macrocells, including violations of the methodology checks. When a problem is identified, it is communicated to the macrocell design team for resolution while chip integration proceeds. When a new type of error is found, enhancements are made to Process Cells so that future chip integration runs can tolerate such an error, and the methodology checks are enhanced to prevent reoccurrence of the error. Hence, each iteration of the methodology results in faster processing of the chip. Once abstracts are generated and the netlist imported into DFII, chip integration can take place. The remainder of the paper details the various chip integration operations summarized in Figure 3

## IV. FLOORPLANNING

During floorplanning, the macrocell blocks (i.e. cache and dataflow macrocells) are pre-placed at fixed locations while the standard cells are optimized with a "pseudo-hierarchical" floorplanning approach.

Pseudo-hierarchical floorplanning is a technique whereby the design can be hierarchically planned and timed without incurring any of the physical design complexities caused by doing true hierarchical floorplanning, placement and routing. It also minimizes potential global suboptimizations due to local optimizations inherent in a hierarchical design approach.

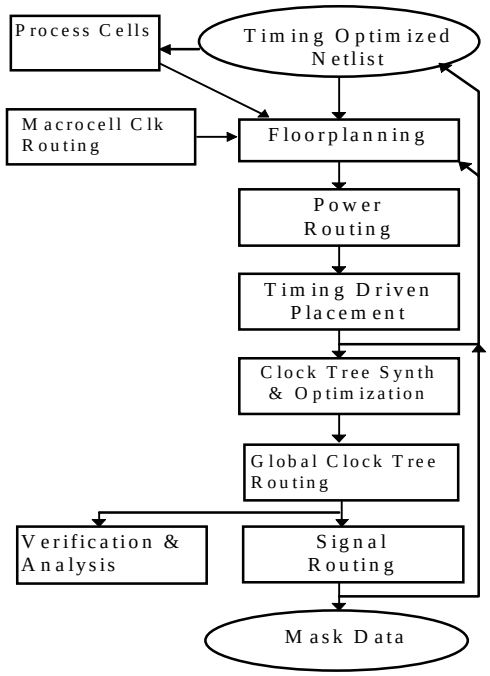


Figure 3 Chip Integration Process

Hierarchical designs have certain advantages over flat designs: early timing prediction, design complexity management, design size management, and concurrent design. However, difficulties arise from concurrent design of the global and lower levels of the hierarchy. Routing resource and area allocations between levels of hierarchy must be carefully and constantly managed to prevent local routing congestion and area growth. Additional problems exist in optimal soft/hard-block pin placement, global area optimization with rectangular soft/hard-blocks, and the timing optimization of critical paths which span multiple levels of the hierarchy.

This pseudo-hierarchical floorplanning approach assigns hierarchical entities of random logic to overlapping regions in the floorplan. The region controls provide a good estimate of the global route lengths for early timing prediction and analysis. Region size, location, and composition are optimized to meet the cycle-time objective (timing closure). Region composition is iteratively determined by analyzing timing constraints and connectivity constraints. Due to tool integration issues, true timing driven floorplanning is not used. Instead, the floorplan is incrementally updated with placement results. The members of each region are color coded so that the floorplanner can graphically analyze the interaction of different regions and interactively modify region size, location, and composition. In this design, 45 logic entities specified in the VHDL are dissolved and reconstituted as 23 tightly integrated regions. Overlapping regions allow cell intermixing between adjacent VHDL logic entities to minimize wire length and congestion. Logic function that is not critical to the timing of the design, such as the self-test control logic, is placed in a region located in the least congested area of the die. Once all logic macrocells have been assigned to regions (more than one macrocell can be assigned to the same region), a flat placement and routing of the entire design can be completed in a matter of a few hours and fed back to the floorplan.

This pseudo-hierarchical floorplanning approach couples the timing control of a hierarchical design with the advantage of a flat design in placing the standard cells in non-rectangular areas. This "pouring" of the standard cells into the natural outline of large pre-designed macrocells yields higher density and performance.

## V. POWER ROUTING

After custom macrocell placement, a power distribution network is algorithmically generated using a custom SKILL program. The power grid is generated on five levels of metal with multiple width wires and pitches supported on each metal level. The algorithm provides the capability to define multiple power terrains by area and level of metal. For example, two areas of the chip may have completely unique grids on all layers. Two other areas of the chip may share a common power grid on metal levels one and two, yet have different power grids on metal levels three though five. Chip-level obstructions can also be added to reserve areas for special routing or wire isolation.

The power grid interaction with a macrocell is personalized for each metal level. Macrocells can interact with a metal level in three ways: stop at the macrocell boundary, wire through the macrocell connecting to power pins and avoiding obstructions, or wire through the macrocell ignoring pins and obstructions. This flexibility allows the power grid to be tailored for each macrocell and assures robust connections.

Power analysis is performed after the grid is personalized. If large IR drops are discovered because of truncation at macrocell boundaries, the underpowered rails are removed from the network. This guarantees that all remaining areas of the power network are adequately connected.

Standard cell core placement sites are now added to the floorplan. By design convention, all standard cells make power connections on the metal-one layer. The algorithm searches the list of all metal one rails and adds core sites in open areas that have the required first metal power rails. The IR drop analysis guarantees that the generated core sites have sufficient power connections and no further analysis is required. The chip is ready for standard cell placement.

## VI. TIMING DRIVEN PLACEMENT

The objective of standard cell placement during floorplanning is to provide accurate feedback to timing analysis for iterative improvement of region composition and macrocell placement. Only region constraints are employed during these

iterations to ensure fast turn-around-time. Once the floorplan is optimized, the placement is further refined by asserting timing constraints to improve critical path delay. This design flow consists of multiple iterations of placement, global routing, parasitic extraction, and timing analysis (Figure 4). During each iteration, the cycle time is measured using the extracted net capacitance and RC. Drive strength of standard cells is readjusted to match the net loading, and the design is re-timed with the newly selected cells. A set of critical paths is identified from this timing run and a new set of tighter capacitance targets is generated to steer placement to a more optimal solution. These steps are repeated until the target cycle time is met or no further improvement is possible. The placement engine, Cadence QPlace™, supports various forms of timing constraints with minimal impact on run time. Net capacitance constraints are applied to nets in the critical paths. This directs QPlace to work harder to minimize these nets, thus improving the chip operating frequency.

The net capacitance targets are generated by invoking a custom program during static timing analysis. Nets in the critical paths are sorted by "slack". Slack is defined as the difference in time between the capture clock and arriving data; positive slack means that the path is faster than the required cycle time while negative slack means that the path is slower. The net capacitance target is set proportional to the path slack, i.e. nets in the paths with more negative slack are assigned a tighter capacitance target. Nets connecting standard cells within a region have a tighter capacitance target than nets connecting standard cells between regions. Nets connecting large custom macrocells are not constrained even if they are in the critical path since they are fixed during floorplanning.

An alternative approach to setting net capacitance targets is to use path constraints, where critical paths are defined with a specified cycle time. In general, this is the best method for timing driven design. However, controlled experiments with path constraints demonstrated that there is only marginal improvement over the net capacitance constraint method for this design. This is due to the iterative refinement of the net capacitance targets over many chip integration passes. Additionally, path constraints are more disruptive to the methodology since the timing analysis tool is not tightly integrated with the placement tool.

The final stage of placement exercises an algorithm which inspects the entire chip for n-well contacts to the power supply. The algorithm optimally inserts n-well contact cells in the standard cell rows to meet the design rule specification.

## VII. CLOCK METHODOLOGY

Clock design is a major part of the overall chip integration methodology and often a critical path to a fast design turn-around time. The hierarchical clock design methodology used here streamlines this process while minimizing clock skew and power.

The system clock distribution network contains three stages, a large global clock buffer (GCB), approximately 20 regional clock buffers (RCBs) and several hundred local clock buffers (LCBs). Control signals on the LCB inputs are used for clock gating and test clock generation. A single phase of the clock is distributed to the inputs of all LCBs. The LCBs generate two pairs of complementary clocks, one pair to the master latches and the other to the slave latches. The portion of clock trees within the custom macrocells are designed and routed as part of the macrocell layouts. The clock interface to a macrocell layout consists of only one input pin. This simplifies global (top-level) clock routing and helps identify inaccessible clock pins. Variable width clock routing of macrocells occurs concurrently with floorplanning so that chip-level information is used to guide the optimization of the macrocell clock tree.

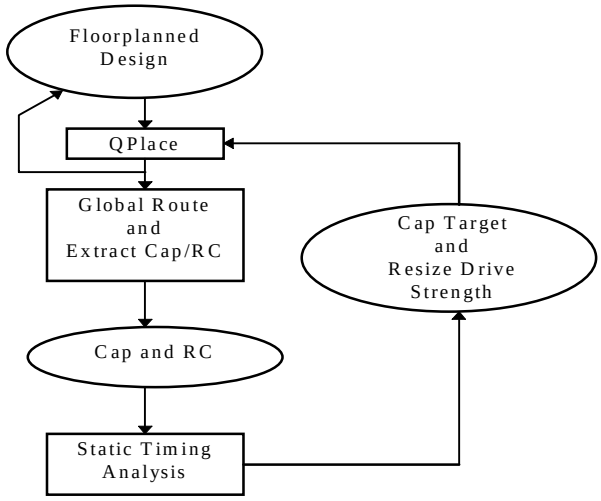


Figure 4 Timing Driven Design Iteration

After floorplanning, placement and power routing are complete, clock tree synthesis and optimization are performed at the global chip level to generate and place clock buffers. This is accomplished using IBM CO2, a tool that traverses the clock trees, identifying and reconfiguring equivalent nets as well as adding parallel copies of buffers to minimize clock skew [1]. The LCBs at the leaf nodes of the clock tree are placed at the RC centroid of the latch cluster to minimize RC induced skews. All the buffers are then snapped to legal placement locations and any cell overlaps resolved. Using IBM CLOCKTREE, a two-layer balanced router that can vary the width of each wire segment, clock routing is then performed on all but the lowest (local clock buffer to latch) levels in the clock tree; the local clock buffer to latch levels are routed with minimum width wires using Cadence Cell3™.

Once the clock design is complete, signal routing takes place. At the same time, the clock nets are extracted, verified and analyzed to ensure the skew objective is met. To ensure a functional design, early mode (fast path) analysis using the extracted SPICE netlist is performed and any problems are fixed either by replacing the LCB with one that has a late launch clock or by adding delays to the fast paths. The capacitance for the entire network including macrocells is 558 pF and the typical clock skew is 180 pS [2]. This compares favorably with the mesh

scheme, where sizable networks having capacitance values of 1400 pF and 2000 pF (for 2 different chips) were required to achieve similar skews [3].

## VIII. ROUTING

Once power routing, placement and clock-tree generation are complete, signal routing is carried out. Because five levels of metal were available for routing, congestion was not a major concern for this design. In addition, it was found through iterative analysis that wiring parasitics extracted after global routing were consistently within 5% of the final routed parasitics, insuring that the cycle time of the routed design could meet the prediction. Therefore, the primary issue during routing was on-chip signal coupling (crosstalk).

Factors contributing to crosstalk include the interaction of five levels of metal for routing, circuits on a net with small output drive devices or pass-gate input structures, fast signal slew rates, and the large die size which leads to long parallel nets with only an 0.9µm separation. Crosstalk avoidance therefore must be considered as part of the overall routing methodology.

One ideal attribute of a router is the ability to route nets such that the simultaneous switching of adjacent signals will not cause signals to fail noise margin guidelines. However, due to existing tool limitations, a modified wide-wire routing technique is adapted to isolate noise-sensitive signals from other nets. This approach is selected because of the availability of routing resources. With simulations and previous hardware indicating that major crosstalk problems occur between signals that are one track away from each other on the same level of metal (for this fabrication technology), noise-sensitive nets are routed first as triple-wide wires. All timing-critical nets are then routed, followed by the remaining nets in the design. The triple-wide wires are then resized back to their default (single) widths, creating an effective isolation region to other nets.

## IX. EC METHODOLOGY

An essential component of the methodology is an EC process for making complex changes to the design without restarting chip integration. An ideal EC process handles an arbitrarily complex change, locates the new logic at the point of failure, impacts a minimum number of manufacturing levels, and does not effect chip cycle-time or area. This methodology addresses all aspects of the ideal EC process, while focusing on two areas in particular:

a) The ability to make complex logic changes using metalization layers only.
b) A technique to add gates as close as possible to connected logic in order to reduce any impact on cycle time.

To facilitate the process, a small EC library consisting of the following gates is created: inverter, NAND, NOR, AND-OR, delay line, terminator (capacitive load for late balancing of clocks), tie-up/down (to tie off gates disabled by an EC), and a latch. These gates are designed with a two-channel gate array primitive (unpersonalized) cell which contains two NMOS and two PMOS transistors, and is personalized only with metal. The EC gates are the same height as, and abut to, the standard cells and can be located in any legal placement site.

At the conclusion of the integration process, at the time the full-chip autoLayout view is converted to a layout view, gate array primitive cells are placed in all unoccupied standard cell placement locations on the chip. EC logic can subsequently be placed at any of these locations.

During final chip integration, one late EC required the transformation of a single cycle path to a half cycle path, the removal of approximately six gates and the addition of approximately twenty gates. The chip cycle time was maintained due to the richness of the EC library and the ability to place the EC elements in close proximity to the change.

## X. DESIGN VERIFICATION

Key to the success of delivering working silicon on an abbreviated schedule is a method to quickly verify the final mask design against the high-level language description of the chip, which is verified by exercising billions of logic simulation cycles. This process is accomplished using formal verification techniques developed at IBM.

Once the VHDL description is verified, a reference model is generated using IBM's logic synthesis system [4]. After synthesis and timing optimization, formal verification is performed against the reference model to verify that logic and timing correction transforms maintained Boolean equivalence with the reference model [5]. This process also verifies that the many logic changes applied during the synthesis and timing correction process are properly incorporated into the design (Figure 5).

After physical design, the resulting schematic netlist is formally verified against the timing-optimized model using the same Boolean equivalency checking software. This process is critical in verifying:

1. Correct operation of data translation from the IBM synthesis system to the Cadence design framework.
2. Logic changes made during the physical design process, such as fanout repowering and clock distribution, are logically equivalent to the original design.
3. Last minute engineering changes (to fix functional bugs or critical path timing problems) are applied correctly.

To complete the formal verification cycle, custom macrocells instantiated in the schematic netlist are formally verified against their VHDL descriptions [6,7].

As a result of these processes and software described above, formal verification insures that the final schematic netlist used for full-chip logic-versus-schematic (LVS) verification is logically equivalent to the VHDL description of the chip.

Due to chip data volume and turnaround time considerations, Avant!™ Vericheck™, a hierarchical physical verification tool for full chip design rule checking (DRC) and LVS checking, is used. Coupled with a macrocell auditing scheme that ensures

DRC and LVS correct standard cells and custom macrocells, hierarchical checking proves extremely advantageous in diagnosing full-chip design errors very quickly.

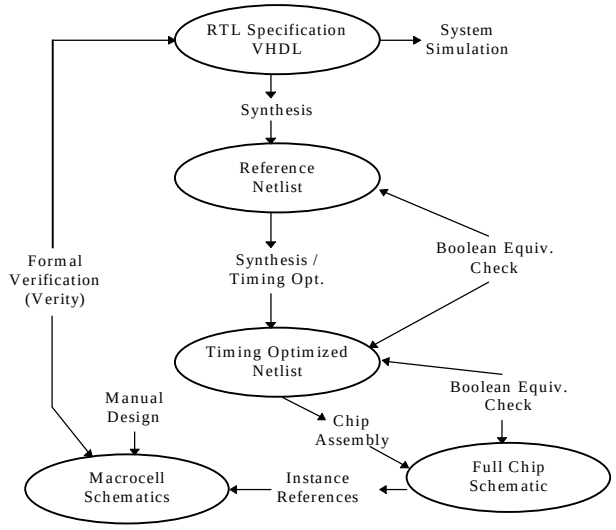


Figure 5 Verification Flow

Critical to the requirement for rapid turnaround time in the "construct by correction" method are two techniques developed to reduce the number of errors caught late in the design cycle.

First, DRC checking is executed on chips of increasing completeness. For example, an initial run is made on a chip that includes only the I/O and the power distribution network. Later, another run is made on a complete floorplan with all custom macrocells placed but with no global routing. With each iteration, more and more of the methodology and floorplan related errors are eliminated.

A second, more important technique is used to provide LVS results on a fully placed, but unrouted chip. Virtual nets are created by attaching text to the terminals of each instance in the design. The text identifies the net that would be attached to each terminal later by the global router. This approach provides critical full-chip LVS results many weeks earlier than the traditional approach.

## XI. SUMMARY

This paper presented a robust design methodology for high performance microprocessors. The methodology focused on "construct by correction" to iteratively improve the design. The pseudo-hierarchical approach provides the benefits of hierarchical design in terms of timing prediction yet suffers none of the shortcomings of hierarchical design management and area suboptimizations. Indeed, the methodology succeeds by postponing critical design decisions to as late in the design cycle as possible. Regions evolve during the design cycle, rather than being predefined at the start, and placement and routing are globally optimized. The methodology was successfully implemented on a PowerPC microprocessor and is easily extensible to other high performance, highly integrated products, such as ASICs.

## XII. ACKNOWLEDGMENTS

The authors would like to acknowledge the contributions of numerous colleagues whose efforts in many diverse areas made this work a success. In particular the authors would like to recognize the efforts of Dave Hathaway, Pete Osler, Betty Bouldin, Tad Wilder, John Doyle, Bruce Winter, Andrew Tickle, Gilles Lamant, Marc Bevis, Andrew Davis, Damian Artt, Don Perley, and Bob Malkemes.



## REFERENCES

[1] D. J. Hathaway, R. R. Habra, E. C. Schanzenbach, and S. J. Rothman, "Circuit Placement, Chip Optimization, and Wire Routing for IBM IC Technology," IBM Journal of Research and Development, vol. 40, no. 4, 1996, pp. 453-460.

[2] K. M. Carrig, A. M. Chu, F. D. Ferraiolo, J. G. Petrovick, P. A. Scott, and R. J. Weiss, "A Clock Methodology for High Performance Microprocessors," to be published in Proceedings of the IEEE Custom Integrated Circuits Conference, May 1997.

[3] M.P. Desai, R. Cvijetic and J. Jensen, "Sizing of Clock Distribution Networks for High Performance CPU Chips," in Proc. Design Automation Conf., June 1996, pp. 389-394.

[4] D. Brand, R. Damiano, L.van Ginniken, and A. Drumm, "In the Driver's Seat of BooleDozer," Proceedings of the IEEE International Conference on Computer-Aided Design, October 1994, pp. 518-521.

[5] D. Kung, R. Damiano, T. Nix, and D. Geiger, "BDDMAP: A Technology Mapper Based on a New Covering Algorithm," Proceedings of the 29th ACM/IEEE Design Automation Conference, June 1992, pp. 484-487.

[6] D. Appenzeller, A Kuehlmann, "Formal Verification of a PowerPC Microprocessor," Proceedings of the IEEE International Conference on Computer Design, October 1995, pp. 79-84.

[7] A. Kuehlmann, A. Srinivasan, and D. LaPotin, "Verity - A Formal Verification Program for Custom CMOS Circuits," IBM Journal of Research and Development, vol. 39, pp. 149-165, January/March 1995.