Design planning for high-performance ASICs

by J. Y. Sayah
R. Gupta
D. D. Sherlekar
P. S. Honsinger
J. M. Apte
S. W. Bollinger
H. H. Chen
S. DasGupta
E. P. Hsieh
A. D. Huber
E. J. Hughes
Z. M. Kurzum
V. B. Rao
T. Tabtieng
V. Valijan
D. Y. Yang

Design planning is emerging as a solution to some of the most difficult challenges of the deep-submicron VLSI design era. Reducing design turnaround time for extremely large designs with ever-increasing clock speeds, while ensuring first-pass implementation success, is exhausting the capabilities of traditional design tools. To solve this problem, we have designed and implemented a hierarchical design planning system that consists of a tightly integrated set of design and analysis tools. The integrated run-time environment, with its rich set of hierarchical, timing-driven design planning and implementation functions, provides an advanced platform for realizing a variety of ASIC and custom methodologies. One of the system's particular strengths is its tight integration with an incremental, static timing engine that assists in achieving timing closure in high-performance designs. The design planner is in production use at IBM internal and at external ASIC design centers.

Introduction

High-performance designs beyond sub-half-micron technologies, with clock frequencies in excess of 100 MHz, have received much attention from the design community and the Electronic Design Automation (EDA) industry in recent years. Interconnect delay has become a significant factor affecting design performance. The key challenges have centered upon managing design complexity and satisfying timing and other design constraints in a small number of design iterations, so as to reduce the overall chip design cycle time and time to market. The traditional approach of sequential, flat physical design (PD), with multiple iterations of placement, routing, and timing verification, can no longer provide an effective chip design solution for dense CMOS technologies. Alternative approaches, based on hierarchical, timing-driven design methodologies that reduce the number of iterations of physical design, are required.

In this paper, we describe an effective approach and solution centered around *design planning*. Design planning is the process of progressively and continuously improving a design concurrently with refining the logic and physical implementations. It is based on analyzing implementation

0018-8646/96/$5.00 © 1996 IBM

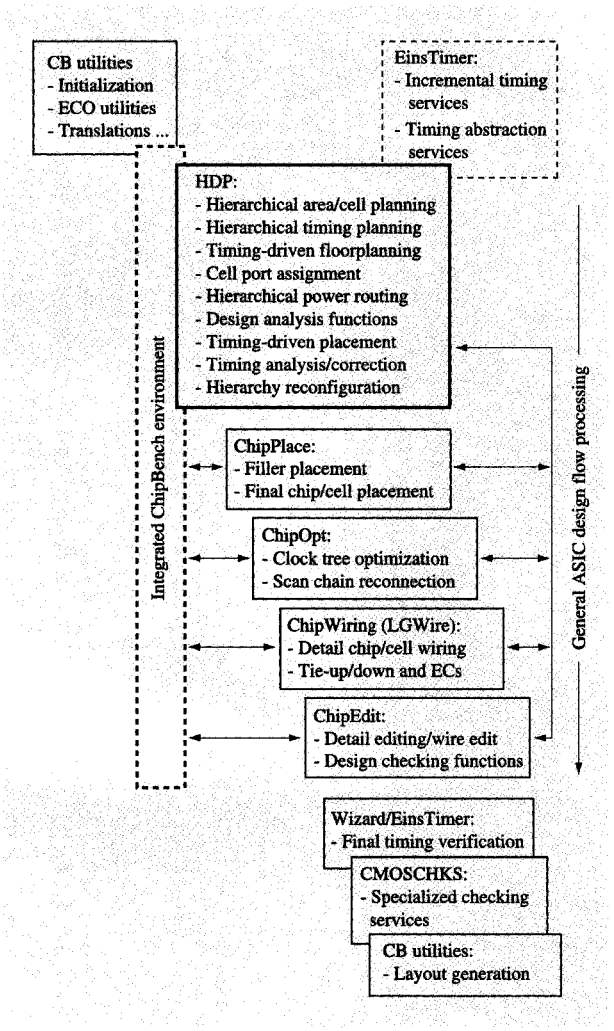

Figure 1

HDP within integrated ChipBench environment in support of high-performance ASIC design.

effects as early in the design decision process as feasible, continually moving the design through successive closure stages. Design planning includes early chip-area planning and analysis, timing/design target generation and budgeting, early and detailed structure planning and floorplanning, wirability and congestion analysis, global routing and wire planning, prerouting *RC* delay estimation, and hierarchical pin assignment.

Beyond the early timing planning of the chip, design planning also drives and optimizes the detailed physical design to satisfy timing and other design constraints. As currently implemented, these advanced capabilities have been used effectively to support a continuum of analysis and design actions, from synthesis to detailed physical design, at one of the most advanced ASIC design settings in the world. First-pass success through automatic placement, routing, and checking has been achieved in many designs with the help of our design planning.

Our solution to design planning has been realized in the *Hierarchical Design Planner* (*HDP*™) environment. The key features implemented in HDP to meet the sub-half-micron design challenges include

- Reflecting physical design and technology constraints early in the design process to minimize the number of design iterations.
- Supporting a seamless, multilevel hierarchy to manage design complexity while preserving the optimization possible with flat designs.
- Supporting improved coupling between the front end and the back end of the design process to ensure close correlation between predicted and actual timing.
- Supporting a mixed level of design detail at any level of the hierarchy during design (physical and timing) planning and analysis to optimize designs that are presented at varying levels of implementation detail.
- Implementing a state-of-the-art timing closure methodology, based on pervasive timing-driven capabilities, to drive rapid timing convergence within the constraints of the design.
- Integrating placement within design planning to support a streamlined timing closure methodology.
- Supporting localized and controlled customization in the ASIC library, such as growable array structures (GRAs) and their customized power supplies.
- Supporting an efficient and granular engineering change methodology that optimizes the changed area while minimizing the perturbation on the physical design of the rest of the chip.
- Improving interoperability with key vendor tools.

HDP can be invoked in a stand-alone mode as an early planner/floorplanner in the logical design environment, or as a hub to manage the full physical design process in the ChipBench™ environment [1]. The ChipBench environment (see **Figures 1** and **2**) is an integrated, hierarchical physical design environment for ASIC and structured-custom methodologies that includes HDP and a number of other tools that make up the environment. Figure 1 highlights the key capabilities of each tool and shows the general ASIC design processing using ChipBench tools in coordination with HDP.

HDP supports integrated early and detailed planning (see **Figure 3**) in any of these configurations. In either setting, HDP analysis and design functions operate seamlessly on hierarchical design elements. HDP provides effective interaction between logical design and physical design, driving PD constraints upward to synthesis and other logic design tools to improve synthesis results, and

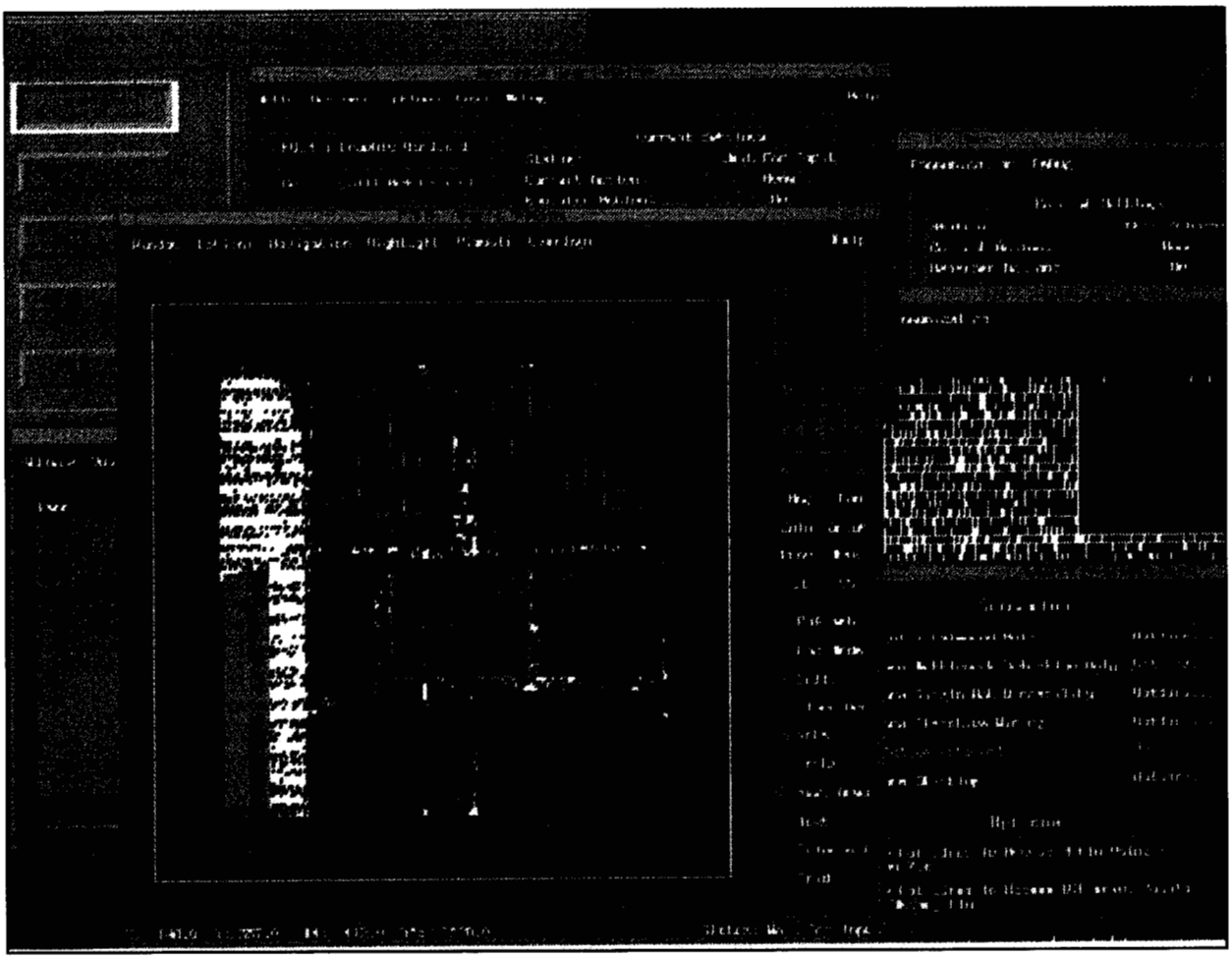

Figure 2

HDP concurrent hierarchical design planning environment.

accepting design targets from logical design as early in the design cycle as design details become available. Operating in the hub mode, HDP provides the components and the execution environment for the timing-driven hierarchical physical design methodology (see **Figure 4**).

A hierarchical timing-driven design methodology, using HDP functions and ChipBench services to implement a complete physical design process, is described in this paper in the section entitled "Performing PD using HDP/ChipBench." HDP supports top-down, middle-out, and bottom-up design implementation methods. It allows the independent implementation of large design entities, as well as the in-context implementation of other entities. Means are provided for encapsulating and isolating hierarchical entities, effectively allowing flat implementation of the overall chip. HDP brings together the components of region placement, target generation and budgeting, and incremental timing analysis to ensure timing closure at the end of a single cycle of placement and routing (see **Figure 5**). The HDP timing closure methodology successively refines the timing targets and implementation constraints, moving the design in a process characterized by rapid timing convergence. During this process, HDP trades off area, wiring length, alternative implementations, wirability, congestion, and other design constraints while incrementally invoking the IBM static timing analysis tool, EinsTimer™ [2], during this convergence process (Figure 1).

The success of this methodology within the context of large hierarchical designs is ensured by a wide repertoire

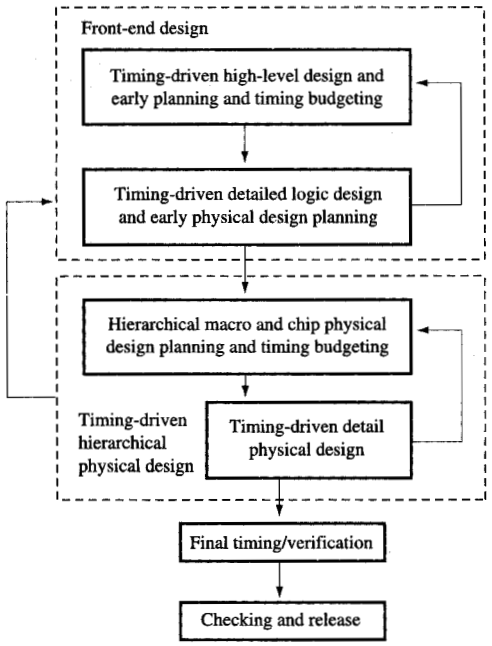

Figure 3

Setting for hierarchical design planning methodology.

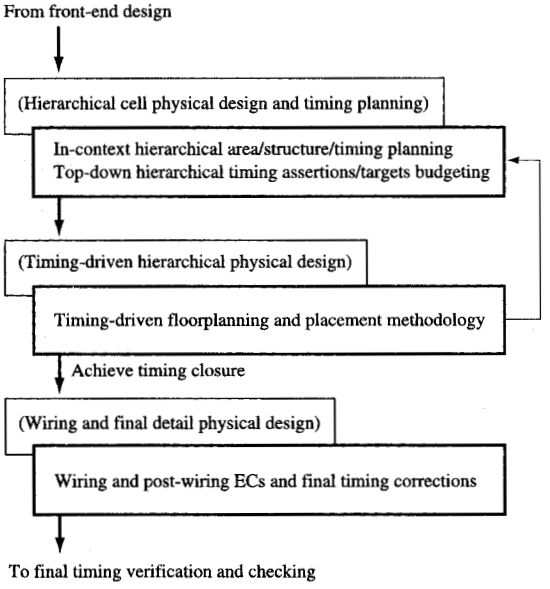

Figure 4

Hierarchical physical design planning.

of ChipBench and HDP services that provide selective, incremental, and on-demand access to a sufficient amount of design and implementation data. This involves the pervasive use of abstractions and controlled access to detailed data as needed, while shielding the user from the burden of explicitly managing the propagation effects across levels of the design hierarchy.

Recently, several DA vendors have released tools to aid in floorplanning and design planning [3, 4]. These tools provide some limited linkages between design functions. The key features that distinguish HDP from those design planning solutions include

- Tight integration of a sophisticated, hierarchical, incremental, and versatile timing-analysis engine. The integration is on-line (i.e., it operates within the same process space).
- Integration of physical design structure planning and timing planning capabilities of preparing physical and timing encapsulation of hierarchical design entities. The process uses selective hierarchy expansion accounting for in-context effects. The quality of results of a subsequent application of the analysis and optimization functions is fully consistent, as if we are operating on the fully expanded design.
- Integration of in-context and encapsulated hierarchical design capabilities, coupled with distributed tool activation, which allows the user to operate in top-down and bottom-up implementation methodologies in the same design environment.
- Integration of hierarchy manipulation and logic optimization services within a physical design environment, which allows the user to restructure and modify the design with instantaneous access to analysis, optimization, and implementation functions.
- Availability of functions that uniformly and transparently work on combinations of available detailed, estimated, or asserted information or actual implementation, providing various degrees of accuracy.

In HDP, a designer can monitor the delay across a critical timing path (see Figure 5) that crosses the hierarchy where some nets are fully wired, some are partially wired, and some entities are abstracted, and some entities are placed. The delay across nets may have been derived using a simple delay model (Elmore delay) or a more accurate evaluation (asymptotic wave evaluation, or AWE). Later, a cell is moved (interactively or by using an integrated automatic function), another cell uses another implementation, and the critical path chart is updated incrementally using EinsTimer static timing analysis. In the meantime, the implementation of a net that was

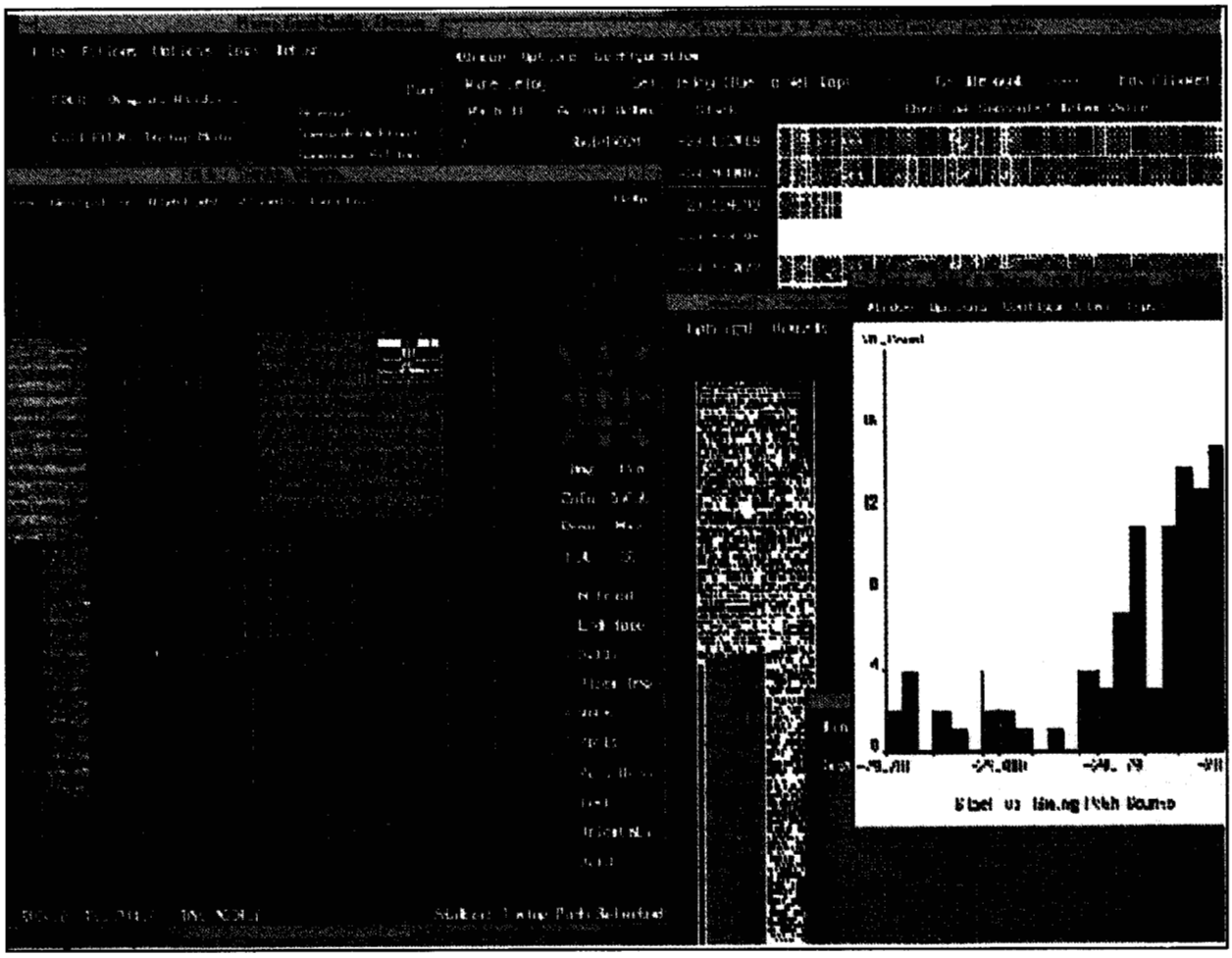

Figure 5

Example of HDP integrated hierarchical timing analysis.

wired automatically is updated for the portions that were affected by the move. Concurrently, one of the hierarchical entities could be under automatic detail placement and another under automatic detail wiring (see Figure 2), while the static timing analysis is using the timing abstraction of these entities. After those actions are completed, the timing abstraction of the modified entities can be regenerated to reflect the latest detail, and the path chart is updated, all without leaving HDP.

The following sections provide details of the capabilities of HDP and the timing closure methodology. There are two design examples that illustrate the timing closure methodology using HDP. Finally, as part of the conclusions, there is a brief description of areas of potential future enhancement.

Functional elements of HDP

This section details the key functional capabilities contained in HDP. All HDP functions are available in a sophisticated multi-windowing graphic user interface (GUI) environment that permits the designer to view, invoke functions, and operate on hierarchical design entities in design context and/or individually. Background activation is available for CPU-intensive applications. Figures 2 and 5 illustrate some of the GUI capabilities of HDP.

One key thread across all functions that deals with the performance elements of a design, whether during analysis, evaluation, or optimization, is the use of the IBM static timing analysis toolkit, EinsTimer. Underlying EinsTimer is a central delay calculation engine, based on the emerging delay calculation language (DCL) standard

[5]. The EinsTimer toolkit contains a rich set of capabilities, such as hierarchical analysis, generation and use of abstractions for hierarchical blocks, and incremental analysis to support small changes in the design. These capabilities are fully integrated in HDP and can be activated as needed during an HDP session.

- *Partitioning*

The hierarchical design capabilities in HDP enable a designer to realize the full potential of the "divide-and-conquer" approach in containing the combinatorial explosion in design complexity that arises from ever-increasing design sizes. These capabilities are further complemented by a rich set of hierarchy manipulation functions. For the purpose of physical design, one can use a hierarchical netlist from the front-end design process, or a reconfigured hierarchy based on physical considerations. Thus, the designer can realize a reduction in design complexity without compromising the ability to optimize and achieve timing closure.

The hierarchy prototyping and reconfiguration facilities in HDP can make changes in hierarchy and effectively manage these changes. The hierarchy creation function creates a new hierarchy specified by the designer or by an application such as the partitioner. The new hierarchy is specified as a set of disjoint clusters of objects. Each multi-object cluster becomes a new block. Single-object clusters are left untouched. The hierarchy flattening function flattens all blocks or a specified subset of blocks in a netlist.

A novel feature of the hierarchy prototyping facilities enables a designer to quickly evaluate alternative hierarchical decompositions without discarding a designated stable hierarchy. At any point, the designer can maintain a stable version of one hierarchy "on the side," while evaluating an alternative "temporary" hierarchy.

The hierarchy manipulation functions are typically needed only if the netlist is very large and flat, or if the use of hierarchy provided by the front-end logical design tools leads to design closure problems.

The partitioning function decomposes a design into a collection of clusters that satisfy minimum and maximum area constraints. These clusters are passed to the hierarchy prototyping services to create a new temporary hierarchy. The function can be invoked at any level of the hierarchy. The partitioner does simultaneous partitioning and floorplanning by default. This permits accurate analysis of global nets, which is essential for accurate timing estimation.

Overall, the principal strength of the partitioner in HDP is its ability to handle multiple performance-related optimization costs and its understanding of physical design constraints. It can handle pre-placed blocks. It can also be used to create an approximate placement of objects in regions corresponding to the partitions, instead of actually creating a new hierarchy. This capability, called region placement, is quite versatile, as evidenced by its various uses in the timing closure methodology described later. It can be used for early placement feedback, as an element of the HDP floorplanning repertoire. It can also serve as the front end of a quick placement program, where the back end performs legalization and overlap removal. Moreover, it is used to find optimized placement of large blocks in netlists that are otherwise flat. This problem is referred to in the literature as the "rocks and sand" problem. The next subsection discusses HDP's main floorplanning engine.

- *Early and detailed floorplanning*

Floorplanning is the element of HDP that interacts with the largest number of optimization, analysis, and planning functions. Additionally, floorplanning addresses problems at various levels of design completion, from early floorplanning through detailed floorplanning and coarse placement.

The engine for automatic floorplan design is based on the simulated annealing algorithm [6, 7]. Annealing is suited for this function, because it is one of the most versatile algorithms for performing "placement-like" optimizations. Its versatility lies in its capability to mix and match objects that are large and small, soft and hard, reshapable and rigid. Additionally, it can be used to optimize any cost function that can be incrementally recalculated quickly.

In exchange for its versatility, the annealing algorithm is relatively time-consuming and complex to control. This is particularly true if the netlist is heterogeneous or contains a wide variety of different objects, or if the cost function contains a large number of different costs. The former problem, that of long run times, is less of an issue in floorplanning than in detailed placement, because floorplanning tends to deal with significantly fewer design objects than does placement. The insights gained from designer experiences can potentially be encapsulated in recipes for controlling the application of the annealing functions.

Interaction with other analysis and optimization tools

The annealing-based floorplanner can move and reshape floorplan entities and, in the process, monitor and optimize a very wide variety of costs. The floorplanner uses a large set of analysis tools to evaluate the cost of a floorplan: overlap computation, overhang computation (representing the overflow of cells outside their parent boundary), wire length estimation, timing analysis and estimation, capacitance estimation, etc. The other optimization tools with which the floorplanner has the most interaction are the partitioner and the pin assignment functions.

Interaction with planning tools

The annealing-based floorplan engine essentially operates on one level of a hierarchical design at a time. Therefore, the timing budgeting, timing target generation, and area planning tools play a crucial role in the success of the floorplanner, in that they reflect the influence of design objects that are deep inside the lower levels of the hierarchy, and they propagate constraints from higher levels in the design hierarchy onto the level being floorplanned.

Support for early floorplanning

The engine for automatic early floorplan design is the same as that in the detailed floorplanner. A primary issue in early floorplan design is to provide the mechanisms for asserting design attributes for unknown portions of the design. The floorplanner in HDP accepts a variety of such asserted design attributes through robust user interfaces and provides capabilities to estimate these design attributes. For example, the designer can specify or estimate sizes and shapes of blocks that have not yet been synthesized, approximate locations of unassigned pins in the netlist using the port area construct, and approximate delay models for large macros whose internals have not yet been designed. Early area and timing planning provide the core functions for these estimation applications.

Support for floorplan editing

All operations that are performed by the automatic floorplanner are available for invocation manually as a floorplan editor. This capability, in conjunction with the rich set of analysis tools available in the HDP environment, enables designers to interactively floorplan complex hierarchical designs.

- *4D planning*

Floorplanning, placement, and wiring tools are specialized for carrying out highly optimized and effective design tasks within the confines of chip or "child" cell boundaries. In a hierarchical design, each of these tasks must be carried out multiple times on different hierarchical entities before the chip design is completed. These hierarchical entities (macro cells) have dependencies on one another—for example, one cell may contain another, one cell may compete with another for area on the chip, or a critical timing path may pass through two macro cells such that there is a constraint on the sum of the path delays through the two cells. Therefore, in order to make the resulting design meet the design requirements, initial design planning work must be done to apportion the design resources (for example, placement area and slots, wiring space, permissible path delays) among the entities in the hierarchy.

The basic hierarchical design planning in HDP is done using two tools: the *area planner* and the *timing planner.* The area planner deals with the three dimensions of cell width, cell height, and wiring layers. The timing planner deals with the fourth dimension of timing and delays.

Area planning

The area planning functions in HDP provide the user with automatic capabilities for apportioning placement and wiring resources among different hierarchical entities. The inputs to area planning are the hierarchical logic netlist of the design, physical data about predefined macro cells and the library elements at the leaf levels of the hierarchy, and technology information such as the number of wiring layers and the total number of available wiring tracks on each layer. There are two kinds of area planning functions, the *early area plan* and the *update area plan.*

The HDP area planner has several enhancements over previously existing area planning approaches found in EDA tools. The most significant is that HDP operates in context within a hierarchical design and understands both aspects of hierarchical design models (full expansion and reuse). In the process, it uses a statistical estimate of wire demand for each hierarchical entity (macro cell) to determine its size, as opposed to an estimate of the "placement" space augmented with an adjustment factor. It automatically analyzes wire demand to determine the number of wiring layers needed for each macro cell, and isolates the wiring problems for different entities from one another rather than using a preassigned wiring layer assignment or depending purely on the user's judgment. It automatically determines a suitable placement grid for each macro cell by analyzing the distribution of sizes of the child cells in the macro, and takes into account this placement grid information in sizing the parent macro cell. It automatically generates shape constraints (aspect ratio bounds) for each macro cell by considering bottom-up as well as top-down constraints. These shape constraints are used locally by the floorplanner to guide it while floorplanning at a particular hierarchy level, without having to access data at other levels.

Early area plan The early area plan function is used to generate initial physical data (macro cell shape, macro cell wiring space reservations, etc.) for a "raw" netlist produced by logic synthesis. It uses information about the library cells and the wiring layers and produces the following outputs:

- *Basic area estimation* A physical shape (width and height) is generated for every macro cell in the hierarchy. The basic area estimation function incorporates methods to compute wire demand estimates for the nets at a certain hierarchy level,

on the basis of a statistical technique for computing average wire lengths.*

- *Wiring area and wire reservations* The area planner creates *wiring-reserved areas* for each macro cell such that the various macro cells in the hierarchy are isolated from one another. This allows the wiring of the nets to be carried out one macro cell at a time in an arbitrary sequence. The wiring tool can work with the blockage and congestion data at each level using an abstracted model of the blockages that are seen at that level.

  A wiring-reserved area is a region of a wiring layer that is reserved for use by the nets of a specified macro cell. This region is seen as free and clear space for wiring the nets owned by that macro cell, but also as a blockage region when nets are being wired for any other cell in the hierarchy.

  Wire reservation determines where wires from a selected parent cell have to use the reserved wiring capacity of its child cells. This is a second-order approximation to that made by area reservation. Area reservation makes the first approximation by determining the area of the cell and its ceiling (representing the space above the cell).
- *Placement constraints* The area planner analyzes the distribution of the shapes and sizes of the library cells that occur in each macro cell, determines an "optimal" periodicity of circuit rows or columns, and creates the placement structures accordingly in each macro cell. This enables the placement functions to operate independently on different macro cells. Note that the placement background is prespecified in certain chip technologies. The area planning functions observe these constraints in placement structure generation.

  The area estimation function described above determines an appropriate width and height for each macro cell in the hierarchy, based on the wire demand in the x and y directions, as well as for any child cells (or other descendants down the hierarchy) that have rigid shapes. However, in order to fit each macro cell and its "sibling" macro cells into the parent macro cell in an effective way, the floorplanning tool must have a range of legal shapes for each of the sibling macro cells. The area planner derives these ranges for each macro cell in the form of "aspect ratio bounds." These aspect ratio bounds, generated by the area planner, have the objective of ensuring that the floorplanner has the maximum legal flexibility in macro cell shapes and that a legal floorplan exists at each hierarchy level.

  Using the number of inputs and outputs of each macro cell, the area planner computes a *wiring buffer* on each edge of the cell. This is a specific number of wiring tracks alongside each edge that are to be kept free and clear by the floorplanner for use as a wiring channel when it places the macro cells within a parent cell. These channels are needed for the wiring program to gain access to the inputs and outputs of each macro cell and to help provide connections between the macro cells. The information about the wiring buffers is provided to the floorplanner as one of the inputs that influence its operation.
- *Area adjustment* In general, the requirements for a chip design are to fit it on the smallest available die size for which there is a chip image in the library. The area planning tool can be run in a mode in which it determines the smallest estimated required size of the top-level cell in the hierarchy (i.e., the whole chip). The user can then select the next largest die size from the library. This results in additional placement and wiring space at the top hierarchy level. The area planner has an area adjustment function that propagates this space down the hierarchy, so that the benefit of the extra space is seen at all hierarchy levels.

Update area plan The update area planning function makes adjustments to the initial area plan using feedback from floorplanning, placement, and wiring. This function performs actions that are similar to those in early area planning; however, they are modified to use feedback rather than to generate initial plans.

Timing planning

The second major component of the design planning capabilities of HDP is the timing planning function. This function deals with planning the fourth dimension of the chip design, delays and timing, across hierarchy levels. The data provided to the function consist of the delay models of the leaf-level elements and the timing assertions (constraints) specified at the primary I/Os at the top level of the design. Using this information, the timing planner

- Apportions the permissible delays across various hierarchy levels, taking into account intrinsic library cell delays as well as net delays.
- Generates timing assertions at the boundaries of the various macro cells and capacitance constraints on the nets across the hierarchy, such that if all of these constraints are met, the timing requirements of the design will be met.

The hierarchical timing planner enables these timing optimization functions by generating the capacitance constraints (or targets) on the nets as well as the timing assertions at the macro cell boundaries, thus allowing the timing-driven floorplanning/placement/wiring runs to be carried out for each macro cell in isolation from the others.

*W. R. Heller, IBM Microelectronics Division, East Fishkill facility, Hopewell Junction, NY, private communications, 1990–1994.

Note that the use of net weights during placement and wiring [8] is perhaps the most popular technique of timing-driven physical design in the EDA industry. Net weights are priority numbers generated for the nets on the basis of their relative timing slack values; these weights are used to influence physical design decisions to help reduce delays on nets that are relatively critical. HDP uses the more powerful technique of net *constraints* rather than net weights to drive physical design, since the former allows a much more accurate and effective fast evaluation of the timing quality of a design, even as the physical design process is being carried out.

The hierarchical timing planning function contains two algorithms described briefly below: *HITARGET* and *HIBUDGET*. The primary difference between them is as follows: The former works across the hierarchy, processing all nets and cells in the hierarchy simultaneously; the latter takes advantage of timing abstractions for the macro cells to process the hierarchy a portion at a time and propagate constraints and estimates across different portions up and down the hierarchy.

HITARGET This is a target-generation algorithm that processes the timing assertions on a design and generates capacitance targets (upper-bound constraints) on the nets in the design such that if the individual capacitance targets are met by placement and wiring, the resulting design will meet the overall timing requirements. Target generation algorithms have been developed and published in the referenced literature [8–10] and are thus not described in detail here. One of the key enhancements in HDP is that it uses a truly hierarchical approach to target generation, in which net capacitances are apportioned not only among different nets at the same hierarchy level, but also across net segments that are connected to one another through ports of macro cells across the hierarchy. An algorithm using a similar approach was recently reported in [11].

HIBUDGET This algorithm, which is an extension of HITARGET, uses timing abstractions to break down a very large hierarchical netlist into portions that can be processed one at a time. It makes use of timing abstractions for macro cells, both to propagate estimated "delay demand" information for macro cells up the hierarchy and to propagate asserted delay constraints down the hierarchy.

- *Integrated net analysis functions*

A net in the HDP environment is basically a collection of ports. In a hierarchical design, a port has two nets—internal and external. An input port is a driver for its internal net and a receiver for its external net. Similarly, an output port is a receiver for its internal net and a driver for its external net. A bidirectional port is both a driver and a receiver for both the internal and the external nets. A routing on a net may be specified by performing global wiring or detailed wiring in HDP, or by loading it from the persistent design database for wires. For the purpose of analysis, the wiring on a previously unrouted net is estimated using a Steiner tree. If a net is only partially wired (i.e., all its ports are not topologically connected), HDP uses "Steiner-like" segments to complete the wiring. The analysis of a net consists of two main steps:

1. *Extraction of the electrical RC network from the physical routing* Each segment in the physical routing is modeled as a distributed *RC* element. A receiver port with no internal net is modeled as a lumped capacitance obtained from the timing (DCL) rule of its cell or from an abstraction. The capacitance for a receiver port with no external net is obtained from the timing assertion constraints used by EinsTimer.
2. *Solving the RC network and storing the results back in the design data model* The effective load presented by an electrical *RC* network on its driver port can be modeled fairly well by a Π-model consisting of two grounded capacitors joined together by a resistor [12]. The solution of the electrical *RC* network for a net at a certain level of the hierarchy begins by computing the Π-model at each of its driver ports, given the Π-model at each of its receiver ports. Therefore, the very first step is to ensure that these Π-models have been computed (or updated) at each of its receiver ports. This is done by "walking through" the hierarchy until a receiver port is reached with either no internal net or no external net. At the request of the user, the *RC* delay, voltage drop, or resistance for each driver–receiver pair of a net may also be computed. The *RC* delay could be a simple Elmore delay [13] or it could be obtained by using the asymptotic waveform evaluation (AWE) method [14]. In this case, every distributed capacitance is replaced by a lumped capacitance of half its value at either of its two nodes in the electrical network.

In addition to the choice of delay computation methods, several options are available to the HDP user for controlling net analysis. The user can select the type of analysis to be performed (*RC* delay, resistance, or voltage drop for each driver–receiver pair), and the mode of analysis (extract the network from its routing, i.e., step 1 above, or model all wires as having zero length, which helps in some cases when the user wants to know how fast the design can run with zero-length interconnections. The user is also given a choice of different Steiner tree heuristics and an "adjust factor," which is used to increase the lengths of all of the Steiner tree segments.

- *Integrated timing analysis*

The primary mechanism for supporting all timing closure activities in the design planner is a set of functions referred to as the *timing services layer*. This layer provides a comprehensive array of services that can be broadly classified into three categories, described in detail below.

Tight integration between HDP and incremental timing analysis

The results of tight integration between HDP and an incremental timing analysis tool (EinsTimer in HDP) may be viewed as a system that consists of two communicating processes cooperating to deliver accurate analysis results whenever physical design changes are made either automatically or manually. For example, when the placement of a cell is changed and the new slack at a port on the cell is requested, this request is passed to EinsTimer by HDP. EinsTimer, in turn, asks HDP to supply the delay values and loading of those interconnections that must be rerouted because of the cell move. These new interconnection delay and loading values are used by EinsTimer, and the new slack is reported back to HDP. This form of tight integration between a PD system and a timing analysis tool has many benefits. From the perspective of facilitating timing closure, the three most important are

- To ensure that the PD environment uses the same yardstick to measure timing closure as is used by the designers of the front-end logical design. This eliminates mismatches in timing analysis results between the front end and PD.
- To permit automation tools and manual changes made in the PD environment to have in-memory interactions with the timer. This permits incremental PD changes to be evaluated using fundamental timing metrics (e.g., slack, arrival time), rather than derived metrics such as capacitances, *RC* delays, and wire lengths.
- As in the first item above, to eliminate mismatches between PD and final sign-off, in a manner similar to that used for IBM CMOS 5 ASICs.

Targeted timing estimation services

Tight integration of the form described above is useful in many situations. However, several automatic optimization methods exist that must support a very high sustained rate of analysis queries. The amount of time spent in inter-tool communication and accurate delay and timing calculation in the above scenario proves to be too expensive for use in the inner loop of these optimization methods. One example of such optimization methods is simulated annealing-based placement. The annealer in HDP makes of the order of 10000 moves per second on a RISC System/6000® Model 590 CPU server. Each move can potentially result in several timing analysis queries. This demanding rate of timing queries is not sustainable by even an extremely fast incremental timer such as EinsTimer. Therefore, the timing services layer in HDP provides some timing estimation tools that can be used in the inner loop of very demanding optimizers such as simulated annealing.

Visualization aids for timing results

Timing analysis typically produces very voluminous timing reports. While these are extremely useful from the perspective of comprehensive analysis of a design, there is a critical need to provide "hot-spot" identification tools, which are embedded in the physical design system and allow the designer to concentrate on problem areas without sifting through substantial amounts of data. HDP provides a very powerful set of such timing visualization tools, which support "timing closure" activities. One example of a visualization tool is illustrated in Figure 5. This tool provides a detailed pictorial breakdown of delays (load-independent cell delay, load-dependent cell delay, interconnection delay), transition times, etc., along the worst-case paths in the design. Each of the delay components is back-annotated to the corresponding design object (cell, net, etc.). A user who wants to perform interactive timing analysis and correction during and after the physical design phase has simply to bring up this tool, click on the offending design entity, and make changes to it in order to correct timing. This feature has proved to be an invaluable aid to designers.

- *In-place optimization*

ASIC design systems that force a complete separation between front-end (logic synthesis) and back-end (physical design) tools almost invariably generate unsatisfiable timing constraints on the back-end tools in the first design pass. This has been the experience of designers of 0.5-μm ASICs, and is likely to continue into the future as fabrication technologies become more aggressive. The primary reason for this situation is that timing constraints generated by the front end and passed down to the back end are based on a timing analysis performed by the front end, using rough or rule-of-thumb estimates of interconnection delay and loading. These estimates are often inaccurate by a large margin because the geometry of nets cannot be accurately predicted before placement is performed. The errors are especially large in ASIC designs (as opposed to structured and hierarchical microprocessor design), because such designs tend to be flat, making it difficult to bound the geometries of nets in the netlist.

Such errors in interconnect estimation were of little consequence in fabrication technologies where the interconnection delay and loading were a very small percentage of the critical path delay. However, in 0.5-μm

technologies, this component of the path delay can be as large as 40% or more of the total path delay. Therefore, timing constraints produced on the basis of the incorrect interconnect estimates tend to yield unattainable timing targets.

One of the techniques for overcoming this problem of incorrect estimations resulting in unattainable timing constraints is to iterate between placement and logic synthesis, and to use the results of placement in one iteration to improve on the estimation and constraint generation for the next iteration. This technique is rather time-consuming and leads to very long ASIC design times, especially for large, flat submicron designs. Additionally, there is no guarantee that the iterations will converge. We have provided an alternative technique for obtaining timing closure in such aggressive designs by integrating some simple logic synthesis operations into the physical design environment. Broadly speaking, all synthesis operations that result in netlist-preserving changes are candidates for integration into the physical design environment. Some examples of candidate operations are gate sizing (also known as power-level optimization) and buffer insertion. Current HDP implementations include gate sizing functions.

Gate sizing in HDP can be invoked manually or by any of an assortment of new automatic heuristics. The automatic heuristics are a combination of local "greedy" approaches and global techniques. The purpose of the automatic function is to resize gates in critical portions of the design in order to eliminate slack, slew, and capacitance limit violations while minimizing placement perturbation. The heuristics are extremely fast, working in a fraction of the time required to iterate between placement and logic synthesis. Our experience has been that they are always successful in eliminating negative slack and capacitance limit violations, and are able to correct a very large percentage of slew violations in the design. Once the resizing is completed, any new resulting physical violations such as overlaps can be removed in the rich physical design environment of HDP.

In summary, by providing the capability to actually make small logic changes in a placed design without leaving the physical design environment, we have been able to achieve timing closure on ASIC designs having constraints that are essentially impossible to satisfy by performing purely physical changes. Designers have found this feature in HDP to be invaluable for meeting timing closure requirements.

- *Global wiring*

Global wiring, in the context of design planning as addressed in this paper, is an analysis tool that creates information used to help make better design decisions. The output of the global router is used to display congestion maps, as well as guide-pin assignment and wire reservation. Global wiring can also be used for early net analysis and timing prediction.

The image on which the router works consists of a checkerboard pattern of tiles on each of the physical layers. Wires, blockage, and capacity are separately accounted for on the boundaries of each tile and within its interior. The representation of these quantities inside the tile as well as on its boundary makes the results of global wiring less sensitive to tile size.

Global wiring consists of several iterations. Each iteration involves evaluating the current solution, selecting nets for rerouting on the basis of that evaluation, and finding alternative global paths for the selected nets. Iterations stop whenever a user-specified limit is reached or the worst local congestion is less than a target congestion.

The first iteration quickly approximates a minimum Steiner tree for all user-selected nets. Layer assignment and L-flipping are performed to even out congestion. The Steiner routing provides an initial estimate of congestion and a basis for subsequently constraining the length of the net relative to its capacitance target. Paths that are contained in tiles or that cross tile boundaries having the worst congestion are selected for reordering during an iteration. These paths are sorted so that paths that tend to occur in regions with a high congestion gradient are visited earlier in an iteration.

The second and subsequent iterations involve rerouting a path using a cost-driven maze runner. Before each path is rerouted, the existing path is used to set cost parameters. Cost parameters are set in order to make finding a path similar to the existing one very expensive and a path that improves on the characteristics of the existing path less expensive. Some of the path characteristics considered are congestion, length, and via count.

When a parent cell has child cells that in turn have hierarchy, a special image is created if global wiring is being used in conjunction with pin assignment or wire reservation. Recall that area reservation is used to determine the area of a cell and its ceiling. The ceiling is used to reserve the wiring capacity of a cell from the first wiring layer to its ceiling. Normally, global wiring at the parent cell would see the reserved wiring capacity of a child cell as a blockage.

Global wiring in HDP offers advantages in that it has the ability to recognize two levels of the hierarchy at a time. This provides more accurate wiring for nets that cross hierarchical boundaries. Furthermore, the global wiring function places wires on individual layers as it proceeds. Since the electrical and geometric properties of wires can vary substantially by layer, this ability provides better planning and estimation capabilities.

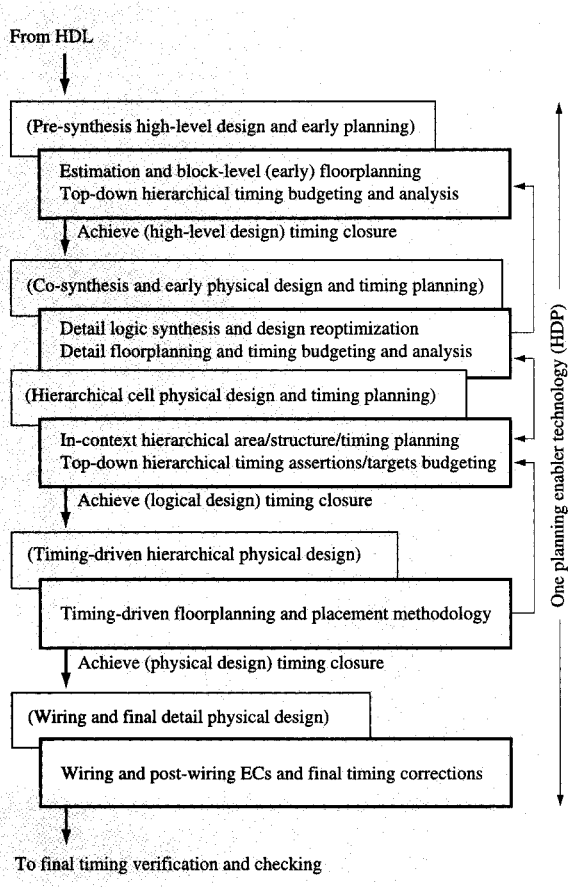

Figure 6

Front-to-back hierarchical design planning methodology.

- *Pin/port assignment*

Detailed macro cell pin assignment determines the locations of pins belonging to cells with hierarchy. Pin assignment can be performed on the pins of the child cells of a selected parent cell or on the pins of the parent cell itself if it is not the chip.

On the child cells, a pin location is chosen to correspond as closely as possible to a point on the boundary of the child cell crossed by previously generated global paths for the net owning the pin. The global paths are optimized to avoid congestion and minimize length. A pin assignment is also chosen to optimize the alignment of pins within the same net.

In addition to pin assignment, there is a placement buffer that expresses an approximation of the free space needed on an outside boundary of a cell in order for its pins to escape. Normally, the placement is close to being finalized before pin assignment, so the buffer is used to make small adjustments only.

Moving beyond these wirability-driven cell port assignments, HDP provides a set of port and I/O assignment functions:

- I/O assignment functions with "legal" placement on I/O drivers, in locations designated for I/O cells on a chip image, if such constraints exist.
- Bottom-up assignment services to allow the propagation of the implementation of child cells.
- Early macro cell pin assignment functions to determine vicinities of cell ports after a floorplanning step. These "port areas" can be used to further constrain the wirability-driven port assignment process.

The advantage of macro pin assignment is again that it has the ability to recognize two levels of hierarchy at a time. The positioning of a macro pin is determined by examining the macro child pins to which a macro pin is connected, as well as the pins of other cells at the same level of hierarchy as the macro. Blockages inside the macro are considered as well as the blockage outside the macro. Other considerations include the width of the wire accessing the pin and the congestion due to other wires in the vicinity of the pin. These considerations provide higher confidence that the pin will be assigned without shorting to a blockage and will be accessible during wiring. In addition, the relationship of the pin with other pins in the net is considered to provide pin alignment.

- *Power and clock routing/planning*

Power routing is used to establish a network of metal interconnections to distribute power from source to sink points. The current methods of establishing this network assume that the designer has some basic pattern in mind for the power network. In practice, this pattern has been a grid of horizontal and vertical lines occurring on horizontal and vertical wiring layers, respectively.

The final phase of power routing is planning. With the pattern as an input, the router modifies the location elements of the pattern with respect to the source and sink points. This is done in a manner not much different than fitting a curve (pattern) to a set of data points (source and sink points).

Once the planning is done, each of the elements of the pattern is implemented to be consistent with a set of constraints. The constraints include limits on the width of the metal, how the metal should be terminated, the distance by which the center of the metal can vary from the center of its pattern element, and the circumstances under which the metal can be interrupted. The pattern is implemented in the presence of blockages with a simple shape-based maze runner.

The implemented pattern comprises the bulk of the metal network of the power distribution system. In practice, this metal network has also been sufficient to

connect all of the source and sink points. If not, the power router offers additional capabilities for connecting any unconnected source and sink points with the metal network corresponding to the pattern.

A ring is a special object that signals different behavior to the power router; it is a rectilinear polygon whose horizontal and vertical elements may occur on different layers. Often, a ring occurs inside a child cell. When this happens, the pattern specified to the power router is interrupted at the ring and not implemented inside the ring. Hence, a ring prevents any power implementation from occurring inside its borders. It also prevents source and sink points existing inside its borders from influencing the power planning.

With hierarchy, the power routing implementation was designed to implement and plan all of the power distribution from the top cell of the hierarchy down to the bottom cells. It is assumed that the same pattern is used throughout the hierarchy. If this is not the case, rings are entered into the cells not following this pattern. Except for connecting into them, no further implementation or planning is done inside these rings or within any other cells occurring inside the rings. After power planning has been done, representations of the power at the top level cell are placed within all ancestor cells, so that design of these cells can proceed independently of their parent cells, but with knowledge of the power at the top level.

The advantage of power routing in HDP is that it recognizes hierarchy in all of its operations and permits approximations of the power to exist during early planning. The power router can implement a complete power network from the chip level that is valid at a given level of the hierarchy. On the other hand, if it is desired to keep the power network outside the boundaries of a cell somewhere in the hierarchy below the chip, it is possible to do so by implementing rings inside the cell. The early approximation of the power permits other planning activities to proceed while taking into account some form of the power network.

ChipBench, through the use of the clock optimization function and HDP, supports clock planning for balanced wiring. The results of the process are fed to the detail wiring program. A companion paper [15] discusses the details of these functions.

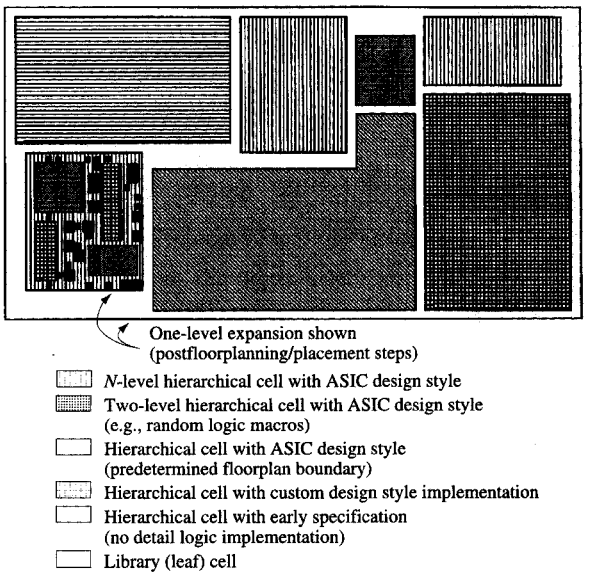

Figure 7

Hierarchical design including entities with mixed-level implementations.

Performing PD using HDP/ChipBench

This section discusses the use of HDP capabilities in a general front-to-back methodology setting (see Figure 3), as well as its application in a physical design setting (see Figure 4).

- *General design planning methodology*

HDP is designed to provide the enabling technology for a general methodology based on timing-driven hierarchical design planning. As shown in **Figure 6**, this begins with a high-level representation of the design, as in structural and behavioral VHDL (verification hardware description language), or Verilog®, and introduces physical design planning early in the design cycle. In this process, HDP is used as an early physical design estimator and floorplanner, interacting with high-level synthesis. For example, in an ASIC design environment, HDP interaction with the main vendor synthesis tool, Synopsys Design Compiler™, is facilitated by the Floorplan Manager™ [16] or an equivalent enabler application. Note that the interaction between HDP and BooleDozer™ [17] (an IBM–EDA synthesis tool), occurs in a natural design flow, with no intermediate hierarchy management facilitator required, since both internal IBM front-end and back-end environments use the same data representation and the common EinsTimer timing engine. As a design progresses, HDP provides planning, analysis, and implementation support for the mixed-level design, incorporating entities that are at different and varying degrees of realization. This assortment of implementation levels, illustrated in **Figure 7**, is handled in a seamless fashion, since the HDP environment is capable of carrying the design to any required level of detailed implementation, including the phases required to generate the actual layout data. This environment also permits the mixing of automatically

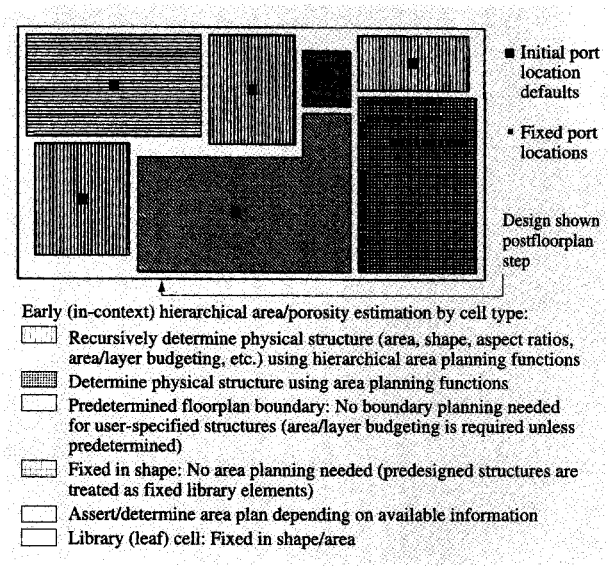

Figure 8

Early in-context hierarchical area and porosity estimation by cell type.

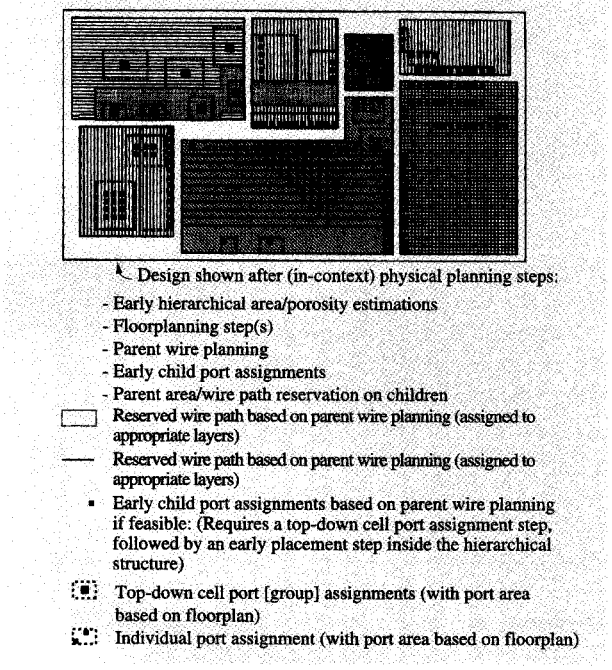

Figure 9

Design after in-context physical planning step sequence.

generated models and asserted (or manually entered) models.

As illustrated in Figure 6, the hierarchical physical design component of the methodology (see Figure 4) and its underlying HDP functional elements are integral to this overall design planning flow. The physical subset of the methodology can be viewed as planning and physical implementation performed in an ASIC supplier design center, while the front-end elements are carried out in a customer design center performing the high-level and logic design. The use of the same planning technology is key to ensuring first-pass success in hardware implementation. We envision that more physical design implementation steps will be executed in ASIC customer design centers to ensure timing closure and shorten the time to market.

Physical and timing planning, as well as the budgeting phases of the methodology, are prevalent throughout the process (see Figures 4 and 5). **Figures 7–11** illustrate the methodology described in this section and show the state of the hierarchical design after the application of the several functions in HDP. Starting with timing constraints at the boundary of a chip (generally in the form of arrival and required arrival times and clocking requirements), the budgeting process allocates delay budgets and provides for propagated assertions to hierarchical child entities at the option of the designer. This process is used in the synthesis domain to drive the detailed netlist generation. In the physical design evaluation or implementation phases, it is used to drive and assist the floorplanning or placement functions. Timing abstractions for any hierarchical design entity can be generated and used in the timing analysis or in timing-driven applications at any stage of the implementation of a design (Figure 11). All processes involving timing are tightly integrated and are based on the same timing technology as that available in EinsTimer.

In parallel, physical design planning uses available early or detailed implementations to allocate physical space to the hierarchical structures. This allocation is three-dimensional, since it involves the allocation of wiring spaces. As an integral part of this wiring area planning, power and clock routing requirements are accounted for, and early implementations are performed as needed. As does its companion timing-budgeting process, the physical-structure planning process can operate with different granularity levels, from asserted space requirements to derived detailed requirements.

The multidimensional physical- and timing-planning processes are used to generate constraints for client applications as well as to define the elements that allow for encapsulated implementations of different hierarchical entities. All planning and encapsulation capabilities are available and operational in the context of the full design hierarchy. This in-context function set includes a full

repertoire of port assignment functions that cover the spectrum from chip I/O assignments to wirability-driven final detail macro assignments (Figure 9). The latter operates in concert with wire- and path-reservation functions, which are also driven by the same global wiring engine. Also, the clock and power structures are planned during the planning processes. The planning of the clock typically involves a different rebuild of the clock tree and results in netlist changes. The planning and implementation of the power supply may involve the activation of signal integrity and verification functions. The clock and power planning capabilities are functions that operate and optimize hierarchically within a design context.

HDP provides an integrated implementation of detailed placement at the detail implementation stage of a hierarchical design entity. The placement and floorplanning methodologies are fully integrated and are supported by the incremental timing capability of EinsTimer, providing active critical path monitoring during the placement process. During the execution of the overall timing-driven, hierarchical, physical design phase of our recommended methodology, clock optimization and other support functions such as scan-chain reconnection can be executed. We also propose the incorporation of data-flow and data-path element support as an integrated extension to the proposed placement methodology.

In the postplacement phase, performance and implementation constraints are generated and passed to ChipBench routing tools or to third-party vendor wiring tools. Throughout the design flow, designers can bring up detail implementations that could have been carried out in an external environment. This includes the accommodation of custom implementations of subsets of the design, as well as accessing electrical analysis results generated outside our tools. Such interoperability in support of this design implementation methodology is facilitated by our repository and by the related environment services. The environment services include ports to existing and emerging industry standards.

Integrated analysis functions, including timing analysis, can be executed at any stage of planning and implementation in HDP. The timing analysis is supported by net analysis functions that take combinations of available implementations, including the automatic utilization of the most detailed section of a hierarchical net, and combine them with the necessary or available estimations, or the global path. For detailed timing verification, EinsTimer can be fed the results of the timing extraction as generated by HDP. In addition to advanced analysis, HDP provides timing correction capabilities (see Figure 5). The HDP functions deal with detailed implementation and mainly involve the repowering of cells, while synthesis actions may involve the resynthesis of a block or a collection of blocks.

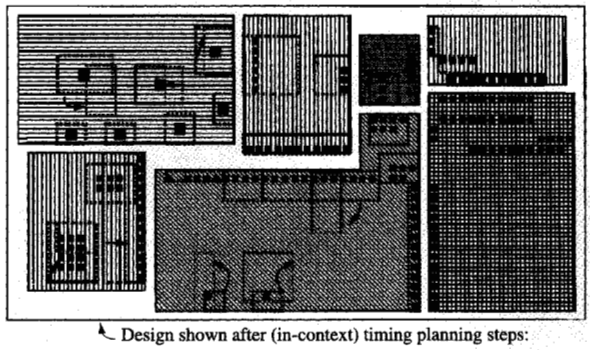

Figure 10

Design after in-context timing planning steps in preparation for final detail PD steps on individual hierarchial entities.

Engineering change order (ECO) capabilities are available and applicable throughout the process. In-place optimizations can effectively take place at any stage of the process. The level of engineering changes that can be supported ranges from the issuance of an ECO command, to automatic detection of a netlist change, to the re-implementation of a large hierarchical entity and its seamless re-imbedding in the rest of the design.

At this point, we take a closer look at the hierarchical placement timing closure methodology, starting with a two-level hierarchical situation.

• *Two-level (flat) placement timing closure methodology*

The placement timing closure methodology provides a general, customizable strategy for achieving timing closure. It has been used with considerable success by HDP users in designing high-performance ASICs, and has improved timing closure and reduced turnaround time by more than 70%. The methodology has been exercised on chips having several million transistors and more than 250000 placeable objects (corresponding to about one million gates), including large blocks such as growable array structures.

The overall methodology consists of three stages. The first stage creates an optimal placement of the large blocks, and derives an effective set of net-capacitance constraints for timing-driven placement. The large blocks

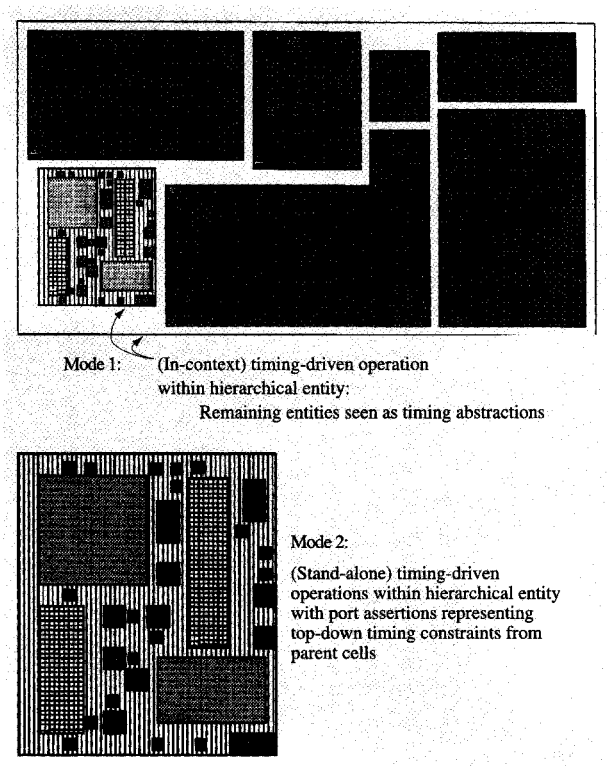

Figure 11

Alternative execution modes of timing-driven operations: in-context execution with possible use of timing abstractions (top) and stand-alone (bottom).

are kept fixed in location during and after this stage. The second stage performs detailed timing-driven placement. The third and final stage performs optimization of scan chains and of the clock-distribution network, along with removal of subsequent overlaps.

The first stage iterates between region placement and net-capacitance target generation. All runs of region placement except the first are timing-driven and use the capacitance targets generated in the previous iteration. The cluster sizes are set to be very small, so the locations are very close to those of an overlap-free placement. The iterations dramatically improve the feasibility of the generated targets, resulting in typical improvements in critical path delays of 10–30% of the cycle time. The strong correlation in timing between the output of region placement and the final overlap-free placement allows one to test for timing closure without removing overlaps. This improves the time per iteration by more than 50%. The timing is usually tested after a trial in-place optimization in HDP has been performed. The iterations stop when timing closure is achieved, or when progress toward it ceases.

The second stage performs low-temperature, simulated-annealing, timing-driven detailed placement on the initial placement from the first stage. This is done in two phases. The first phase uses only net-capacitance constraints and restricts the range of movement of objects from their initial location. This helps minimize overlaps while maintaining the optimization achieved in the first stage. The second phase runs without these move constraints, but with capacitance constraints as well as critical paths. This phase marks several short annealing runs interspersed with the updating of timing-critical paths. This typically improves the delay on the worst path by another 10%.

The final stage performs optimizations of the scan chains and of the clock distribution network. This is followed by removal of any overlaps created by clock optimization, which is done with a very low-temperature simulated-annealing run of timing-driven detailed placement to preserve timing closure.

- *Hierarchical timing closure methodology*

HDP allows for a wide spectrum of choices for processing the design, ranging from completely flat placement and wiring at one extreme to processing across multiple levels of hierarchy at the other. If a particular design is small enough to be processed comfortably without partitioning, the designer may choose to process the design flat; in this case, the placement methodology of 4D planning is applied.

However, in many situations a hierarchical approach is more effective, and sometimes it is necessary. The criteria include the size of the design, the mix of critical and noncritical components, the level of reuse of high-level components, and the existence of a mixture of design styles across different design partitions. For example, if some partitions contain random standard logic while others contain structured-custom logic, it may be more effective and practical to apply different placement algorithms to different portions.

The placement timing closure methodology can be readily adapted to any level of the hierarchy, using the region placement, floorplanning, or detailed placement function with the hierarchical timing planning functions described in the subsection on 4D planning. This extension forms the core top-down hierarchical timing closure methodology. The last stage of the placement methodology, which performs scan-chain and clock optimization, is applied only at the very end. The modified methodology using the floorplanner is described below. If placement constraints such as row structures are to be observed, the detailed placement tool is used. Region placement may be used in the first stage if the design contains a large number of small cells, with only a few large unplaced blocks.

Top-down hierarchical timing closure methodology
The modified, top-down methodology consists of the following steps:

1. If the design contains only leaf-level elements, apply the placement timing closure methodology.
2. Find a good set of targets: Iterate between quick floorplanning and timing planning to generate capacitance/*RC* targets on nets and delay assertions on hierarchical blocks, until timing closure is achieved or progress toward it ceases. All but the first floorplanning runs are timing-driven, based on net-capacitance targets. These runs are kept short by permitting overlaps, as long as the overlaps are small.
3. Perform the final timing-driven floorplanning to create an overlap-free floorplan. Use both timing-driven features, and perform several short, low-temperature runs interspersed with the updating of critical paths. The floorplanner also creates port areas for unassigned macro ports to drive the design at the next level. Finish with timing planning to refine timing budgets and create boundary timing assertions for all child blocks.
4. Recursively apply the top-down hierarchical timing closure methodology to every child macro.

The overall hierarchical timing closure methodology is as follows:

1. Perform the following preprocessing steps:
   a. Starting with the logical hierarchy, carry out automatic or manual repartitioning of the design (if desired) to obtain a physical hierarchy (see the subsection on partitioning).
   b. Perform quick floorplanning and placement over the hierarchy (this step is optional; it provides a better starting point for timing planning).
2. Perform bottom-up timing abstraction generation (see the subsections on 4D planning and integrated timing analysis).
3. Apply the top-down hierarchical timing closure methodology. If timing closure is achieved, exit.
4. If significant progress has not been made toward timing closure, selectively flatten the problem blocks, and either repartition them or leave them flat. Return to step 3 (in all cases).

• *Summary of HDP timing closure methodology*
The driving factors in our proposed design methodology are physical design and timing closure. HDP brings together the components of physical structure planning, floorplanning and detailed placement, target generation/budgeting, and incremental timing analysis to achieve timing closure. By being able to operate on mixed-level designs, and by taking advantage of logic estimation and emerging front-end floorplan data management technologies, we are effectively blurring the artificial boundaries between physical design and high-level design.

The goal of this timing closure methodology is to achieve closure by the end of the placement phase, so that first-pass routing success can be effectively achieved. HDP timing closure methodology successively refines the timing targets and implementation constraints, moving the design in a process characterized by rapid timing convergence. HDP timing closure methodology trades off area, wiring length, alternative implementation, wirability, congestion, and other design constraints while invoking EinsTimer incrementally during this convergence process. As design entities are sent to a wiring tool that observes the constraints generated by HDP, the wired designs are guaranteed to satisfy the timing and electrical requirements.

The success of this methodology within the context of large designs is ensured by a wide repertoire of system services. In HDP, the basic mode of operation is selective and incremental access to a sufficient amount of design and implementation data on demand. The use of abstractions and controlled access to detailed data is achieved while shielding the user from the burden of explicitly managing the propagation of implementation effects across levels of the design hierarchy.

• *Concurrent launching of back-end PD processes*
One of the key components in supporting an advanced design planning methodology is to allow the concurrent activation of functions on hierarchical entities. In a typical utilization scenario, a designer is capable of starting (from within HDP) back-end placement and wiring functions as separate processes on entities below the chip level. This permits concurrent, side-by-side design on the same chip by one or more designers. All of the tools shown in Figures 1 and 2 can be activated concurrently if the state of the design permits. HDP and the integrated ChipBench environment manage all required locking, data integrity controls, and any necessary data exchanges among interacting tools. Furthermore, all data propagation among hierarchical design entities is performed without intervention from the designer. Future extensions of the environment services will provide the user with more scenario and control capabilities, allowing for further exploitation of the network-centric computing paradigm.

• *Structured-custom extensions*
The ChipBench environment offers structured-custom extensions which provide an active link between ChipBench tools (primarily through HDP) and IBM–EDA CircuitBench™ tools (primarily through the layout-editing environment GYM). Passive links with third-party vendor tools are available through a GDSII-based interface that

can be complemented by constraints exchange using the Physical Design Language (PDL) and the ECO facilities available in ChipBench. Both elements of bottom-up (full-custom) and top-down (semi-custom) design methodologies are supported in the current release of ChipBench. Physical design constraints are exchanged between the interacting tools. ChipBench supports continuous activation and interaction between HDP and GYM. HDP can currently pass outline and pin location constraints to GYM. Physical design abstractions are automatically generated, allowing the seamless integration between editing a library entity and applying place and route tools on its parent.

Design examples using HDP timing closure methodology

The effectiveness of our flat timing closure methodology in achieving timing closure on large, flat timing-critical designs is demonstrated in this section with two typical designs that were successfully completed with the methodology. Examples 1 and 2 below track the successive improvement in the worst timing slack in the designs as they proceeded through the process.

- *Design example 1*
- 1.5M-transistor design (CMOS 5L).
- 26 large GRAs automatically placed.
- 480 I/Os.
- 10-ns cycle time.
- Two iterations of region placement and target generation followed by timing-driven floorplanning/placement, in-place optimization, overlap removal, and user-assisted path fixing.

	Timing slack improvement over previous step (ns)
Net-length-driven	—
Cap-target-driven (all nets) + target generation	1.7
Critical timing paths + cap-target-driven	1.0
In-place optimization + overlap removal	1.8
User-assisted path fixing	0.3
Total improvement in timing	4.8 ns (meets timing requirement)

- *Design example 2*
- 1M-transistor design (CMOS 5L). 32 large GRAs automatically placed.
- 300 I/Os.
- 17-ns cycle time.
- Three iterations of region placement and target generation followed by timing-driven floorplanning/placement, in-place optimization, and overlap removal.

	Timing slack improvement over previous step (ns)
Net-length-driven	—
Cap-target-driven (all nets) + target generation	1.3
Critical timing paths + cap-target-driven	0.6
In-place optimization + overlap removal	1.4
Total improvement in timing	3.3 ns (meets timing requirement)

The examples show that our tightly integrated timing-driven capabilities lead to substantial improvements over net-length-driven physical design. Furthermore, the integration of these capabilities in one design environment permits the user to perform successive timing closure steps without having to incur the overhead of switching tools or execution domains.

Traditional synthesis bases its predictions of timing on statistical connection-length estimates. While such estimates may be accurate on the average, they have potential for leading to poor timing estimates. This phenomenon was not evident until designs became large while feature sizes became smaller and wire-dependent delays became a factor. Our experience has been that the slack predicted by synthesis is generally overly optimistic. Accounting for the effects of physical design early in the design process should substantially reduce the potential discrepancy between predicted and achievable timing.

The majority of the designs that have been processed to date through HDP and ChipBench have used a flat PD methodology, primarily because of our system's ability to process a flat physical design efficiently and the growing popularity of the ASIC-plus-cores design style. However, as design sizes and functions have grown and the need for planning the design in the front-end domain has increased for high-end ASICs, we are witnessing more designs that must be processed hierarchically in PD.

Summary and future directions

In this paper we have described the Hierarchical Design Planner (HDP), which provides a combined environment for hierarchical physical design and for both interactive and automatic design planning. HDP includes a partitioner that can create a physical hierarchy from a design that either is flat or has a logical hierarchy, and a floorplanner with many advanced features. Both are tightly integrated with EinsTimer to provide true timing-driven capabilities. They are capable of handling many other constraints to

make both the partitioner and the floorplanner truly versatile tools. HDP also comes with advanced checking capabilities and a global router that provides valuable feedback to the floorplanner and also drives detailed routing. Because of the tight integration of floorplanning and timing, HDP provides a natural environment to achieve timing closure through quick iterations of design scenarios in complex designs.

Using HDP within advanced ASIC design centers, to activate and control a truly integrated timing-driven methodology in the setting of IBM ChipBench for physical design, has led to substantial reductions in design time for timing-critical applications. The paper has presented some design examples in which large improvements were achieved in terms of timing over traditional net-length-based physical design optimization. More significantly, this was accomplished without having to perform synthesis or return to a front-end design environment.

Looking into the future, we see cores and embedded macros as one of the key challenges. While HDP handles macros today, we expect their type and complexity to increase in step with the development of new technologies. Designs with complex cores are just beginning to emerge. Because they now appear as "black boxes," the challenge from a floorplanning point of view will be to extract enough timing (path and pin-to-pin delays) and physical data (pin locations, feed-throughs, blockages) to ensure that the cores can be placed amidst their surrounding logic. But that is only the beginning. Soon, cores may be described only logically or abstractly, and their physical and possibly detailed logical design may have to be done along with the rest of the logic, but with particular restrictions (size, area, shape, all with limited knowledge of their internal design) that pertain to their internal characteristics and periphery.

Another key area of enhancement involves linkage with behavioral-level design and synthesis. While links exist to back-annotate timing and some physical data, the future requires design planning to be tied to the behavioral and architectural levels of design, where estimates of timing of critical paths and physical shapes and sizes will allow rapid estimation of performance, cost, and size. At the architectural level, feedback on physical data will also allow the system designer to decide the "cut-points" between hardware and software in a hardware–software co-design environment.

Acknowledgments

The authors wish to express their appreciation to J. Darringer, G. W. Doerre, and W. R. Kehrli for their encouragement and support. We would also like to extend our thanks and appreciation to the rest of the ChipBench, EinsTimer development, and EDA support teams.

References

1. "ChipBench at a Glance," *IBM–EDA Document 0020-5272*, IBM Microelectronics Division, Hopewell Junction, NY, April 1995.
2. "EinsTimer Users' Guide and Language Reference," *IBM–EDA Document 0020-5261*, IBM Microelectronics Division, Hopewell Junction, NY, March 1996.
3. "PREVIEW," *http://www.cadence.com/preview.html*, Cadence Design Systems, Inc., San Jose, CA 95134, 1996.
4. "Design Planner 3.2, Reference," HDL Systems, Inc., Santa Clara, CA 95054, 1995.
5. IEEE Standards Committee, "Common Delay Calculator User's Guide and Language Reference Manual" (draft of DCL language), IEEE, Piscataway, NJ, 1995.
6. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulating Annealing," *Science* **220**, 671–680 (1983).
7. C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package," *IEEE J. Solid-State Circuits* **SC-20**, No. 2, 510–522 (1985).
8. W. K. Luk, "A Fast Physical Constraint Generator for Timing-Driven Layout," *Proceedings of the 28th ACM/IEEE Design Automation Conference*, June 1991, pp. 626–631.
9. Peter S. Hauge, Ravi Nair, and Ellen J. Yoffa, "Circuit Placement for Predictable Performance," *Proceedings of the IEEE International Conference on Computer-Aided Design*, Santa Clara, CA, November 1987, pp. 88–91.
10. H. Youssef and E. Shragowitz, "Timing Constraints for Correct Performance," *Proceedings of the IEEE International Conference on Computer-Aided Design*, November 1990, pp. 24–27.
11. S. V. Venkatesh, "Hierarchical Timing-Driven Floorplanning and Place and Route Using a Timing Budgeter," *Proceedings of the Custom Integrated Circuits Conference*, May 1995, pp. 469–472.
12. P. R. O'Brien and T. L. Savarino, "Modeling the Driven-Point Characteristic of Resistive Interconnect for Accurate Delay Estimation," *Proceedings of the IEEE International Conference on Computer-Aided Design*, November 1989, pp. 24–27.
13. W. C. Elmore, "The Transient Response of Damped Linear Networks with Particular Regard to Wideband Amplifiers," *J. Appl. Phys.* **19**, 55–63 (1948).
14. L. T. Pillage and R. A. Rohrer, "Asymptotic Waveform Evaluation for Timing Analysis," *IEEE Trans. Computer-Aided Design* **9**, 352–366 (1990).
15. D. J. Hathaway, R. R. Habra, E. C. Schanzenbach, and S. J. Rothman, "Circuit Placement, Chip Optimization, and Wire Routing," *IBM J. Res. Develop.* **40**, 453–460 (1996, this issue).
16. N. Deo, *Deep Submicron Technology Design Methodology Backgrounder*, Synopsys, Inc., Mountain View, CA 94043, June 1995.
17. L. Stok, D. S. Kung, D. Brand, A. D. Drumm, A. J. Sullivan, L. N. Reddy, N. Hieter, D. J. Geiger, H. H. Chao, and P. J. Osler, "BooleDozer: Logic Synthesis for ASICs," *IBM J. Res. Develop.* **40**, No. 4, 407–430 (1996, this issue).

Received November 7, 1995; accepted for publication April 11, 1996

John Y. Sayah *IBM Microelectronics Division, Route 52, Hopewell Junction, New York 12533 (sayah@vnet.ibm.com).*
Dr. Sayah is a senior engineer/scientist at IBM Microelectronics. He is the lead architect, principal designer, and overall technical project lead of the Hierarchical Design Planner (HDP) and ChipBench products in support of advanced hierarchical VLSI design for deep-submicron technologies. Dr. Sayah received a B.Sc. degree in physics from the Faculty of Sciences at the Lebanese University–Beirut, an M.Sc. degree in electrical and electronics engineering from King's College at the University of London–UK, and a Ph.D. degree in electrical and computer engineering from the University of Wisconsin at Madison.

Rajesh Gupta *IBM Microelectronics Division, Route 52, Hopewell Junction, New York 12533 (rgupta@vnet.ibm.com).*
Dr. Gupta is an advisory engineer/scientist at IBM Microelectronics. He is the technical project leader responsible for the hierarchical planning functions and timing-driven applications support in HDP. Dr. Gupta received the B.Tech. degree in electronics engineering from the Indian Institute of Technology, Madras, in 1985. He received the M.S. and Ph.D. degrees in computer engineering from the University of Southern California, Los Angeles, in 1987 and 1991, respectively. In 1979 he was awarded the National Talent Scholarship by the Government of India. His interests include various aspects of VLSI design and test, particularly timing-driven physical design, hierarchical CAD frameworks, and design for testability.

Deepak D. Sherlekar *IBM Microelectronics Division, Route 52, Hopewell Junction, New York 12533 (deepak@vnet.ibm.com).*
Dr. Sherlekar is an advisory engineer/scientist at IBM Microelectronics. He is the technical project leader responsible for the partitioning, floorplanning, and placement functions in HDP. He received a B.E. degree in electronics engineering from the University of Indore–India, an M.Tech. degree in computer science from the Indian Institute of Technology–Kanpur, and a Ph.D. degree in computer science from the University of Maryland–College Park. Prior to joining IBM, he was an assistant professor of electrical engineering and computer science at the University of Michigan–Ann Arbor. His interests include design automation for high-performance VLSI, algorithms for combinatorial optimization, and parallel computing.

Philip S. Honsinger *IBM Microelectronics Division, Route 52, Hopewell Junction, New York 12533 (honsinge@vnet.ibm.com).*
Dr. Honsinger is a senior engineer/scientist at IBM Microelectronics. He is the technical project leader responsible for chip wiring, which includes power routing, global routing, detail routing, and macro pin assignment. Dr. Honsinger received his Ph.D. degree in 1978 in high-energy particle physics from Ohio University. He spent two years as an assistant professor at West Virginia Wesleyan College, returning to Ohio State University to obtain an M.S. degree in computer science in 1981. Dr. Honsinger has been at IBM since 1982, working in EDA in the area of physical design. He has extensive experience in high-performance wiring for packaging, including considerations for noise and signal integrity issues. He has coauthored several papers and holds several patents.

Jitendra M. Apte *AT&T Laboratories, 101 Crawfords Corner Road, Holmdel, New Jersey 07733.*
Dr. Apte is currently working in the area of on-line services at AT&T Laboratories, where he is pursuing various projects related to Internet-based electronic commerce, as well as interactive and multimedia services on the World Wide Web. Before joining AT&T Laboratories, he worked in the Electronic Design Automation Laboratory at IBM Microelectronics. There he was involved in the development of algorithms and software for several design planning problems: specifically, automatic floorplanning design, timing-driven design, and interactions between logic/HL synthesis and design planning. Prior to his work at IBM, Dr. Apte was involved in the development of automation tools for physical design in the Design Technology Laboratory of the Hewlett-Packard Company. There he implemented floorplanning tools that were used in the design of VLSI chips for manufacturing popular HP products, including laser printers and high-performance workstations. Dr. Apte's current technical interests span a wide range of topics, including on-line services that use real-time video and audio content, secure-payment mechanisms for supporting electronic commerce on the World Wide Web, development of interactive client-executable Web content using emerging languages such as Java, JavaScript, and VRML, as well as issues in automatic design and analysis of hardware and software systems. He received a bachelor's degree in electrical engineering from the Indian Institute of Technology, Bombay, in 1985, and a Ph.D. degree in computer science from Duke University in 1990.

S. Wayne Bollinger *IBM Microelectronics Division, Route 52, Hopewell Junction, New York 12533 (bollinge@vnet.ibm.com).*
Dr. Bollinger is an advisory engineer/scientist at IBM Microelectronics. He joined IBM in 1992 and is actively involved in the development of physical design applications, providing technical direction in the areas of user interface, intertool communication, and application integration and architecture. His interests include object-oriented analysis and design, software engineering, parallel and distributed computing, design and optimization of VLSI circuits, and switch-level test generation. Dr. Bollinger received B.S., M.S., and Ph.D. degrees in electrical engineering from Virginia Polytechnic Institute and State University.

Hai Hsia Chen *IBM Microelectronics Division, Route 52, Hopewell Junction, New York 12533 (CHEN at FSHVM1).*
Ms. Chen is part of the EDA group at IBM Microelectronics; she has been with IBM since 1982. Her experience includes work on routers for modules, cards, and boards, and she is currently working on the power bus router. Ms. Chen received a B.S. degree in physics from Taiwan Normal University in 1973 and an M.S. degree in computer science from Rensselaer Polytechnic Institute in 1982.

Sumit DasGupta *SEMATECH Corporation, 2706 Montopolis Drive, Austin, Texas 78741.* Dr. DasGupta is currently on assignment from IBM at SEMATECH, where he is the program manager for design for test and physical design. Prior to that, he spent more than twenty years in electronic design automation at IBM, with the greatest emphasis on physical design and design for test. His last assignment in EDA was as project manager for chip physical design. Dr. DasGupta has earned the fourth-level Invention Achievement Award at IBM. He has published more than fifteen papers in external conferences and journals and has received several awards for his activities in IEEE, of which he is a senior member. Among his past and present activities with IEEE, he is a past Chair, and current member, of the Steering Committee for the *IEEE Transactions on VLSI,* past Editor-in-Chief of the *IEEE Design and Test of Computers* magazine, and past General Chair of the International Conference on Computer Design. Dr. DasGupta received a B.S. degree in electrical engineering from Jadavpur University, Calcutta, India, an M.S. degree in electrical engineering from Marquette University, and a Ph.D. degree in computer science from Syracuse University.

Edward P. Hsieh *IBM Microelectronics Division, Route 52, Hopewell Junction, New York 12533 (hsiehe@vnet.ibm.com).* Dr. Hsieh is a senior engineering manager at IBM Microelectronics. He has more than twenty years' experience in electronic design automation at IBM, in the areas of VLSI chip and package physical design, test design automation, and technology/manufacturing automation. He currently manages the development of advanced chip design tool capabilities in support of deep-submicron technologies, including ChipBench and Hierarchical Design Planner products. Dr. Hsieh received a B.S. degree in electrical engineering from the National Taiwan University, an M.S. degree in electrical engineering from MIT, and a Ph.D. degree in electrical engineering and computer science from Columbia University. He has served on the Technical Committee of ICCD since 1990 in the capacities of session chairs and vice chairman for design and test, and he served as a technical committee member for MCMC.

Andrew D. Huber *IBM Microelectronics Division, Route 52, Hopewell Junction, New York 12533 (ahuber@vnet.ibm.com).* Mr. Huber is an advisory programmer at IBM Microelectronics. He is currently responsible for technical coordination of EDA chip physical design product releases. He has recently been involved in software development and providing technical direction in the areas of system and data model support for the chip hierarchical physical design system. Prior to that, he worked in the area of automatic wiring tools, making significant contributions to the automatic routing software for printed circuit boards, MCMs, and TCMs. Mr. Huber also managed the EDS Chip Technology Department, directing the development of chip automatic placement and wiring tools. He received a bachelor's degree in computer science from Michigan State University in 1979.

Edward J. Hughes *IBM Microelectronics Division, Route 52, Hopewell Junction, New York 12533 (hughese@vnet.ibm.com).* Mr. Hughes is a staff programmer at IBM Microelectronics in the Electronic Design Automation laboratory. Here he concentrates on physical design with a focus on user interface and GUI. Since 1989, he has worked in an object-oriented environment on multiple interactive application projects. He is currently working on an interactive application architecture toolkit with a built-in command language. Mr. Hughes received a B.S. degree in information systems from the University of Scranton in 1989.

Zahi M. Kurzum *IBM Microelectronics Division, Route 52, Hopewell Junction, New York 12533 (zkurzum@vnet.ibm.com).* Mr. Kurzum is a staff development engineer at IBM Microelectronics. Joining IBM in 1989, he worked in the area of physical design, making a significant contribution to the development of global wiring and hierarchical pin assignment. He is currently working on detailed placement for hierarchical designs. Mr. Kurzum received his B.S. and M.S. degrees in computer engineering from Clemson University in 1984 and 1989, respectively. He is a member of Tau Beta Pi.

Vasant B. Rao *IBM Microelectronics Division, Route 52, Hopewell Junction, New York 12533 (raov@vnet.ibm.com).* Dr. Rao received his B.Tech. degree in electrical engineering (electronics) from the Indian Institute of Technology, Madras, in 1980, and his M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana–Champaign in 1982 and 1985, respectively. In 1985, he joined the faculty in the Department of Electrical and Computer Engineering at the University of Illinois at Urbana–Champaign as an assistant professor. In the summer of 1987, he worked as a consultant with Texas Instruments, Dallas. Dr. Rao joined the IBM EDA Laboratories in East Fishkill, New York, in 1991; his research interests have included static timing analysis, physical design, simulation and optimization of VLSI circuits, stochastic algorithms for combinatorial optimization, and graph theory. He has published more than fifty papers in various journals and conference proceedings.

Thepthai Tabtieng *IBM Microelectronics Division, Route 52, Hopewell Junction, New York 12533 (tabtieng@vnet.ibm.com).* Mr. Tabtieng is a staff engineer at IBM Microelectronics. He joined IBM in 1990 and is working on the development of EDA physical design tools. He received a B.S. degree in computer engineering from Northeastern University in 1985 and an M.S. degree in electrical engineering from Tufts University in 1990. Mr. Tabtieng is a member of IEEE and the IEEE Computer Society.

Vigen Valijan *IBM Microelectronics Division, Route 52, Hopewell Junction, New York 12533 (valijan@vnet.ibm.com).* Mr. Valijan is an advisory programmer at IBM Microelectronics. He joined IBM in 1984 and has worked on various GUI applications in the field of semiconductor design (chips, cards, etc.) on MVS and AIX® platforms. Since 1990, he has been a major contributor to an interactive application architecture that provides an object-oriented paradigm for multiwindow, multithreaded GUI applications using

X Windows®/Motif® toolkits on AIX. Mr. Valijan received a B.A. degree in mathematics and computer science from Queens College, CUNY, in 1982, and an M.S. degree in computer science from Marist College in 1988. He is a member of Pi Mu Epsilon.

David Y. Yang *IBM Microelectronics Division, Route 52, Hopewell Junction, New York 12533 (DAVID_ YANG@vnet.ibm.com).* Mr. Yang is a staff development programmer at IBM Microelectronics. He joined IBM in 1990 and is currently working on ChipBench environment and model services, in the area of system interfaces and data models. Mr. Yang received a B.E. degree in computer applications from Beijing Iron and Steel Institute, Beijing, People's Republic of China, in 1982, and an M.A. degree in computer science from Queens College, CUNY, in 1990.