Unifying Multiple Tools to Achieve High Performance SoC Design

Mark Bales, Joe Mastroianni, David Gregory, Paul Rodman

ReShape, Inc., Mountain View, CA, USA

Abstract

This paper describes the challenges facing IC implementation today, and concludes that a multi-vendor EDA system is a necessity in today's complex design environment. We introduce the concept of Chip-Level Design Automation (CLDA), the next level of abstraction in physical design, which relies to a much greater degree on automation in order to leave the designer free to consider higher-level design issues and gives the designer more time to explore the full implementation design space. Arguments are made that claim CLDA can provide a 10X productivity improvement, and actual measurements using the ReShape software system validate the claims, and show CLDA to be a new level of abstraction worthy of attention.

Introduction

The "design productivity gap" is defined as the gulf between what is possible to manufacture and what is possible to design by the ITRS [ITRS99]. Ron Collett, of Numetrics Corporation, documented the rising SoC design productivity gap at the Electronic Design Processes Workshop in 2001 [Collet01]. In 2003, International Business Strategies, Inc. [IBS03] showed that while costs were increasing, physical design productivity has been decreasing since the 0.35-micron technology (See Figure 1).

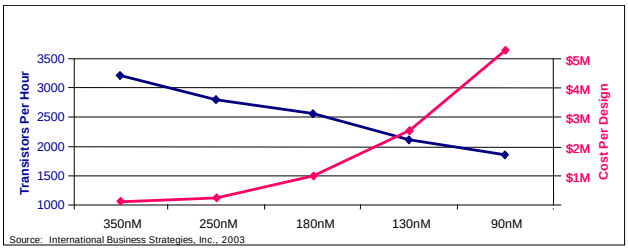

Figure 1 Chip design cost vs. Design productivity trend

Many factors contribute to this effect. Exponential growth in transistor complexity is compounded with growing interconnect parameter complexity and increasing operating frequencies for designs. What were previously interconnect parasitics are now primary effects. Clock frequencies are already in the multi-GHz range for CPUs and network chips, and are approaching the GHz range in graphics and other high-end chips. Designers must augment their work flow with implementation and verification steps to fully manage these effects. Yet, fundamental implementation steps are still dependent on 20-year-old design automation technology.

Cell-based place and route (P&R) has supported a 1,000x complexity growth since the mid-1980s. While the EDA industry has invested heavily to maintain the effectiveness of cell-level P&R it is falling behind in its ability to keep up with the design challenges.
Product price/performance ratios exert incredible pressure on Quality of Results (QoR). Viability can be gained or lost in slim margins even before technology pressures are considered. Mobile computing challenges power consumption in designs, complexity places demands on manufacturability and design cycle, and short product life-cycles pressure time-to-market and reduced NRE costs, which aim straight toward engineering productivity.
Use of hierarchy is no longer a convenience, but a necessity. Eighty percent of today's hierarchical design effort is focused on block specification (pins, repeaters, constraints), running the tools, and integration verification. Only twenty percent of the effort is spent on full-chip global decisions like chip area, global interconnect, power, and clocking. Making these decisions without knowing how the implementation will

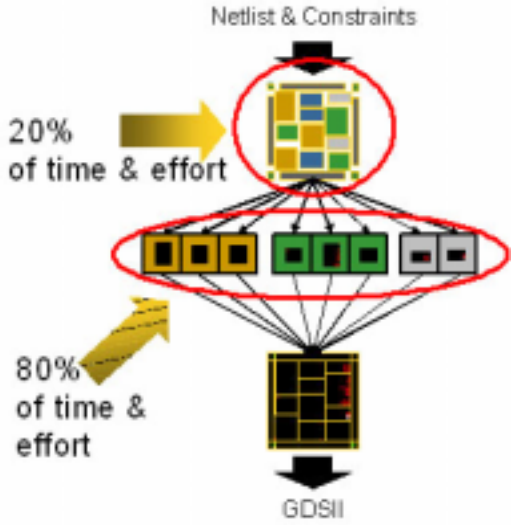

Figure 2 Systems where most time is spent in blocks

affect the design results in ineffective iteration. Because hierarchy cannot be properly managed, most designs are locked into a frozen set of blocks far too early in the design process.
Chip-Level Design Automation (CLDA) is aimed at enabling SoC design teams to focus on higher levels of

implementation while increasing the "communication" with automated block-implementation tasks.

Challenges in IC Physical Implementation

Flow not an integrated part of the design data

Logic designers capture their design intent by writing RTL and specifying design constraints. They use synthesis technology to bind function (specified by the RTL), performance (design constraints) with a cell-library (e.g. Artisan, LSI Logic, etc.)

By contrast, physical designers express their design intent through a floorplan and a series of tool sequences and tool commands. These are captured in tool command files, UNIX scripts, and make files that often total hundreds of thousands of lines for complex chips. They are typically written as one-off design-dependent programs hardwiring aspects of the netlist, floorplan, IP, the versions of the tools being used, and the tool settings.

There are three problems with the scripted design approach. First, managing and maintaining script code complexity is labor intensive. It is a fact of life in physical design that netlist, floorplan, IP, tool versions, and tool settings change nearly to the minute the "tape" ships to the mask shop, and change breaks the hard-coded scripts.

Second, these "scripted specifications" are generally incomplete and require significant manual intervention to produce mask-ready layout. For example, some or all of these issues require manual oversight:

- Manual data preparation
- Monitoring script progress
- Recovery from scripting errors and tool failures
- Machine and license resource management
- Determining next step after a script completes
- Script maintenance
- Design-specific optimizations

Finally, because the team objectives are hardware, rather than tool related, time pressures make it impractical to generalize the scripts as reusable tools. This is analogous to front end design where writing reusable RTL requires twice the effort of implementing the original function. Compounded by the fact that physical design teams are always asked to make up for the delays that were incurred upstream, this negates any hope of script reuse across design organizations, and sometimes even between designs.

Single Vendor Solutions Insufficient

It is tempting to consider a single-vendor solution to these flow issues. Most single-vendor solutions are well integrated, and some today are quite comprehensive. Irrespective of any vendor's propensity to accrete technology through merger or acquisition the fact remains that no single vendor is the best at everything. In addition, EDA technology leadership is transitory as competitors leapfrog the older systems and evolving process nodes drive new technical solutions.

We posit that given any design flow, every single-vendor solution to that flow is missing perceived vital construction or verification technology. In addition, the best-integrated single-vendor solutions will be the hardest to extend in "non-standard" ways by design. While an end-user company with a large central CAD group may bear the overhead to extend such systems, it is almost impossible for a circuit design group to accomplish this without paid consulting help. Even in the cases where the single-vendor provides consulting help, extension can be problematic if the extension involves a 3rd-party tool from a competing vendor.

As mentioned before, a successful flow's relevance may still be transient in nature. It is likely that such a flow will work for a given design, but it is also likely that it will require modification from one design to the next, and it is almost a certainty that it will require modification from one process node to the next. The chance of having all the right tools available for these cases from a single vendor is extremely low.

The larger EDA companies which have grown through acquisition have problems providing an ultimately unified solution with acquired technology. There are instances where 3rd-party companies are able to more fully integrate two EDA tools that belong to the same large company, but most often the job of integration is left to the end-user group or to their central CAD department. This is the source of the oft-quoted customer lament that $3-5 is spent on integration for every $1 spent on EDA tool purchases.

Challenge is to manage uncertainty

Add to these challenges that the blocks in the design may be finished at different rates. Complex EDA tools may display stochastic behavior, affecting the overall convergence of the design. An overabundance of logistical issues is present in getting different tools to interoperate. Interaction between blocks and between the blocks and top-level chip attributes must be considered while the design progresses. Finally, the design work may be spread out over groups in geographically disparate locations. The challenge in IC design is to obtain the adequate process observability and controllability to manage uncertainty and drive productivity.

Chip-Level Design Automation

The Next 10X in Productivity

Chip-level Design Automation is the next level of design abstraction for physical designers. With CLDA, the designers' attention rises to optimizing whole-chip issues from the very start of the design cycle. Designers make decisions in the context of the full chip, and get verification feedback from mask-ready layout to those decisions in unprecedented turnaround times.

Making Hierarchy Work

The issues involved in handling hierarchical design and the inadequacy of hierarchical design tools has been a barrier to most teams attempting to use this design style. One key to making it work is total automation of the chip construction process. If the full-chip layout can be iterated quickly, chip-level issues surface quickly.

This level of automation is powered by combining three flow-process techniques: 1) a replayable design specification, 2) splitting the flow into modular, plug and play construction steps, and 3) flow elaboration (i.e. creating customized scripts for the specific chip) at runtime.

1) CLDA assumes the physical designer implements the SoC function defined by the RTL designers, within timing and cost (area) constraints in an environment of constant change. A resilient physical design specification is needed to make change management easy. With CLDA a SoC can be reconstructed without designers having to get into the details of individual cell, macro or pre-route placements, or how the netlist interacts with the design specification.
2) To fully automate the design flow, the flow is broken up into units of physical design work, called stages. Stages are high-level construction steps such as place, global route, and clock tree synthesis. By connecting stages together, designers can customize a block construction recipe in a systematic way (tool settings, for example.) Stages have error checking and recovery to stop construction if errors occur, and recover from the basic failure modes.

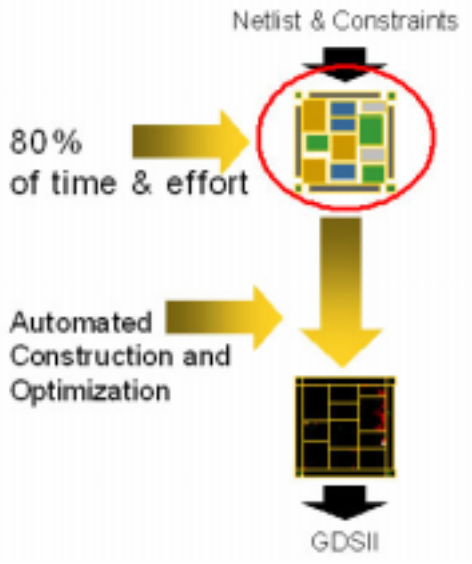

Figure 3 CLDA lets designers spend most time on global issues

3) Flow elaboration at runtime generates custom tool scripts for each block in context of the design. The elaboration process takes the physical design intent as specified by the floorplan, tool sequence (specified from a library of flow stages as described above), and tool settings specified for each block in the design—combined with the logic designers intent (netlist and constraints)—to generate the hundreds of thousands of lines of customized scripts for the specific design. The elaboration process allows user input to be largely abstracted from the details of specific tools and netlists, even though the resulting tool command files and scripts are specific to them. Physical design teams enjoy a 100X reduction in the volume of script coding they must manage.

CLDA provides teams with full visibility into the status of flow completion coupled with data-mining to provide a complete snap-shot of overall design status at any point in time. This allows design management to clearly pinpoint progress, to apportion resources, and to identify roadblocks in chip implementation.

Because the chip is implemented quickly, CLDA optimizations improve quality of results for chip area, performance, and power consumption. Place and route algorithms are non-deterministic and exhibit stochastic behavior. As a consequence design teams today invest a tremendous amount of effort in determining optimal pin locations on blocks, placement of global repeaters/buffers, and propagation of Synopsys design constraints (SDCs) onto the individual blocks. CLDA automatic optimizations abstract away the low-level details of individual block specification for place and route.

CLDA QoR

For CLDA to be accepted there can be no degradation in QoR over existing design methods. CLDA technology improves chip quality in three ways:

1. Fast design iterations enable engineers to explore and verify many more design options using the production place and route and verification tools so there is no miscorrelation of estimation models, because the behavior of the P&R tools is used directly to drive design decisions.
2. Abutted block hierarchical design style enables high-performance, high-volume microprocessor and graphic processor designs, and has been found to decrease die sizes by 10 to 20% over conventional channeled routing schemes. Professor Cong at UCLA reports that breaking designs into smaller blocks (250K instances, or about one million gates) delivers higher quality

than large flat place and route techniques [Cong01].

3. Design Aware™ technology when combined with abutted methodology yields a quality of results identical to flat design. The user pays no penalty for hierarchical partitioning. The Design Aware algorithm uses the results of prior chip layout decisions to drive new layout iterations just as a designer does by hand. ReShape CLDA uses Design Aware optimization for pin placement and global signal repeater insertion, resulting in global wire length reductions of 30 to 40 percent, with a corresponding improvement in the timing and power consumption of these signals (see Figure 5). A mixture of Design-Aware and estimated information may be used, so convergence can still be achieved even in the event blocks are finished at different points in time.

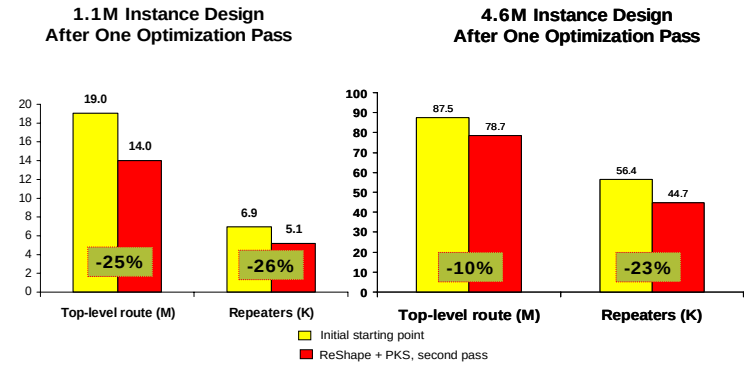

Figure 5 Effects of Design-Aware Optimization

PD Planner

This is the front end of the ReShape design system. It is the floorplanning system in which a user creates a replayable floorplan script. In the figure below is displayed a chip floorplan including the detail of the pads, the top-level (soft) blocks, macro cells within the top-level blocks, and repeaters that have been introduced to buffer top-level nets. The original gate-level netlist has been repartitioned. Not visible but present are power rings and meshes for the chip, and the logical implementation for the clock tree. In addition to the ability to create replayable floorplans, PD Planner includes basic floorplanning capabilities such as pad placement, grouping and repartitioning, power planning and analysis, clock-tree synthesis, timing budgeting, top-block and macro-block placement, and global signal repeatering.

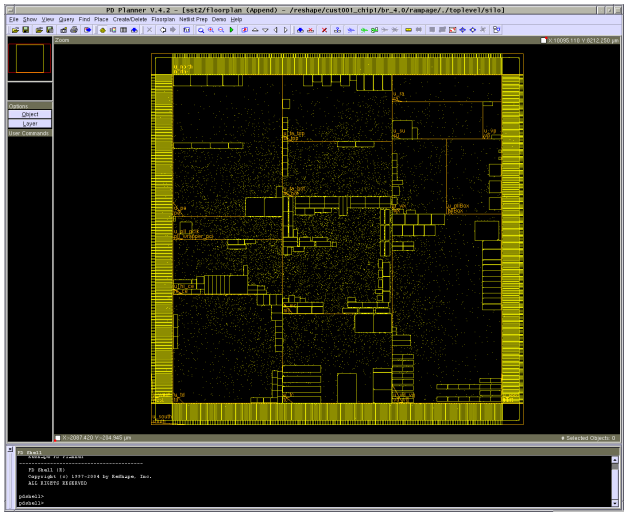

Figure 5 PD Planner window with Replayable Floorplan

PD Planner helps the user concentrate on the high-level aspects of design, leaving the construction of the blocks to the 3rd-party tools driven with PD Builder (described below). PD Planner uses an abutted methodology enabled by powerful optimization technologies and provides Virtual Flat results.

PD Builder

PD Builder is the back end of the ReShape design system. It is the control system that drives implementation of the blocks through the 3rd-party tools. In the picture shown below, you can see the spreadsheet-style interface. The rows represent stages of the flow, and each column represents a block in the design. The flows are customizable in many ways. The simplest way the flow can be changed is to choose a default flow for the chip. Currently, timed and untimed variations are available for Synopsys- and Cadence-based tool flows.

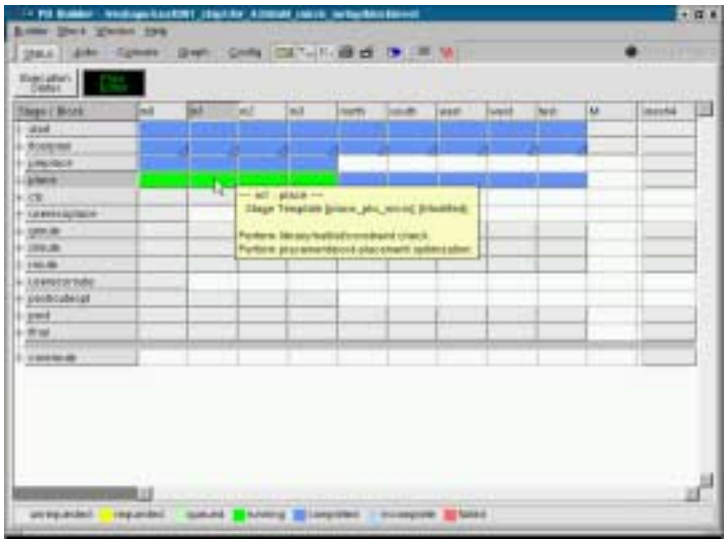

Figure 4 PD Builder window building chip automatically

OpenFlows

Once a flow is chosen, each step or stage within the flow can be parameterized through changing predefined *vars* that control the execution of a given stage. It is possible to set a different subflow for each block. For example, in the figure above, block *m0* could be using a Cadence flow and block *m1* could be using a Synopsys flow. This

allows for a form of "what if" exploration to minimize each block.

In addition to customizing each block with its own flow, it is possible to customize the flows with user-written stages that can add other 3rd-party or proprietary tools to the flows driven through PD Builder.

Results

Three important scenarios are used to measure physical design productivity:

The number of lines of code it takes to describe an SoC

The time to assemble the initial full-chip implementation

The time to assemble a revised, incremental floorplan and netlist

First, most physical design team create 100's of thousands lines of script code to codify the construction of a SoC. Because CLDA abstracts away the binding of physical design intent from the netlist, technology, tools, there is typically a 100X reduction in scripts required to specify a SoC in a CLDA system vs. a cell-based system—a productivity improvement of at least 10X.

Second, a team of eight designers takes six to nine weeks to assemble and perform the initial build of a hierarchical SoC with eight to 12 place and route blocks and one

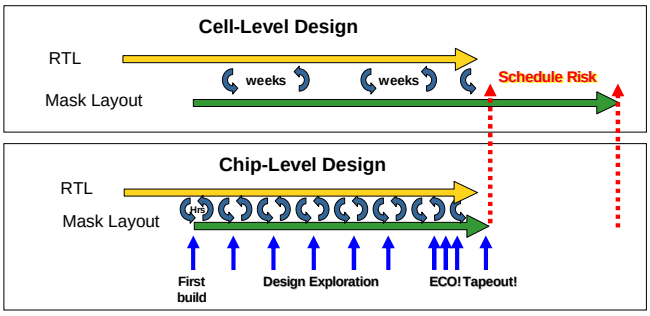

Figure 6 Shorter Iteration times means more iteration

million placeable instances. (This is a big reason why full-chip analysis is often deferred to late in the design cycle.) Comparable designs using ReShape tools take two engineers just two weeks. The productivity gain for a first build is at least 10X.

Third, and perhaps most important, is the time it takes to support a design engineering change order, ECO. This is the critical metric, because it enables efficient design exploration in the early part of the design stage, and

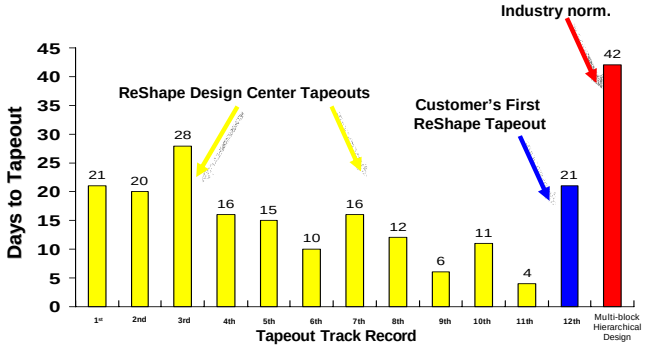

Figure 7 Actual tool tapeout times from final netlist

accommodates the inevitable last-minute design changes imposed by (1) the system verification uncovering bugs, (2) the customer, or (3) the market. Using ReShape's CLDA tools, one engineer can build a completely new design from a new floorplan and netlist in 24 hours. A traditional tool methodology will take the eight-person team from five to 14 days, depending on the extent of the changes. The incremental productivity boost is at least 40X. It is this productivity boost that makes full-chip physical design exploration, and dealing effectively with late breaking ECOs practical. The bottom line is higher quality chips on schedule. See Figures 7 and 8.

References

[Collet01] "Key Performance Indicators of Methodology Capabilities," presentation by Ron Collett, Numetrics Corporation, at the Electronic Design Process Workshop, 2001, Foil 14, "ASSP Project Distribution by Design Productivity, Design Capacity & Development Cost." http://www.eda.org/edps/edp01/SLIDES/collett.ppt

[Cong03] "Optimal Scalability Study of Existing Placement Algorithms," Cong, J, et al, Asia South Pacific DAC, Asia South Pacific Design Automation Conference, Kitakyushu, Japan, pp. 621-627, January 2003.

[IBS03] "Analysis of the Relationship between EDA Expenditures and Competitive Positioning of IC Vendors, a custom study for the EDA Consortium," International Business Strategies, Inc., 2003. Table 1.3, page 14, Head Counts and Design Costs. http://www.edac.org/downloads/resources/profitability/HandelJonesReport.pdf

[ITRS99] "Design Productivity Gap," Figure 5, page 6, International Technology Roadmap for Semiconductors, 1999 edition. http://public.itrs.net/files/1999_SIA_Roadmap/Design.pdf