

IC Compiler II: Finding the Best Floorplan, Fast

Meeting the Increasing Challenges of Today's and Tomorrow's System-on-Chip (SoC) Hierarchical Designs

Author

Steve Kister

Technical Marketing Manager,
Synopsys, Inc.

Floor Plan Exploration Challenges

As designers strive to pack more and more functionality into today's SoC's, design size (in terms of the number of transistors packed into a chip) is growing almost exponentially. This growth brings with it an unbounded increase in not just the technical complexity of performing the physical layout of the design due to capacity challenges, but also requires designers to make choices that can have profound effects on a design's ultimate tractability and marketability.

For these large, complex designs, a flat layout methodology is no longer an option. Prohibitive implementation runtime coupled with high memory requirements means that these designs have to be implemented hierarchically in a divide and conquer methodology.

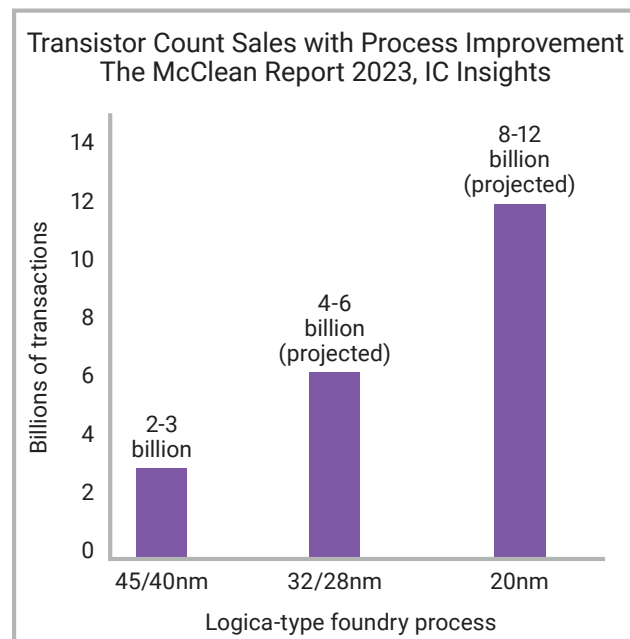


Figure 1: Increasing Transistor Counts of Designs

This divide-and-conquer approach allows teams to partition designs into hierarchical sub-blocks, often divided along functional lines—for example CPUs, GPUs—thus helping to ease the complexity of the problem as a whole. These sub-blocks can be implemented in parallel amongst a broader design team resulting in faster total design closure.

It is in the preparation for this hierarchical implementation phase when physical design teams must start by exploring floorplan solutions. This task is often carried out using early design data that merely estimates the physical aspects of the design. Based on this very early, usually incomplete data, designers must somehow quickly determine if the competing goals of area and operating speed can be achieved while ensuring that the power footprint can concurrently be met. In this phase design teams must strive to find solutions to problems that are more difficult to recover from if discovered later during the implementation phase of the flow.

Due to schedule constraints, a generally short period of time is allocated for designers to explore initial, but often design bounding, floorplan solutions. For hierarchical designs, individuals or teams need to be enabled to quickly create top-level and block level floorplans to allow parallel assessment of solutions across a breadth of costs. The faster they can create and assess floorplans, the more potential solutions they can consider within this short, scheduled time.

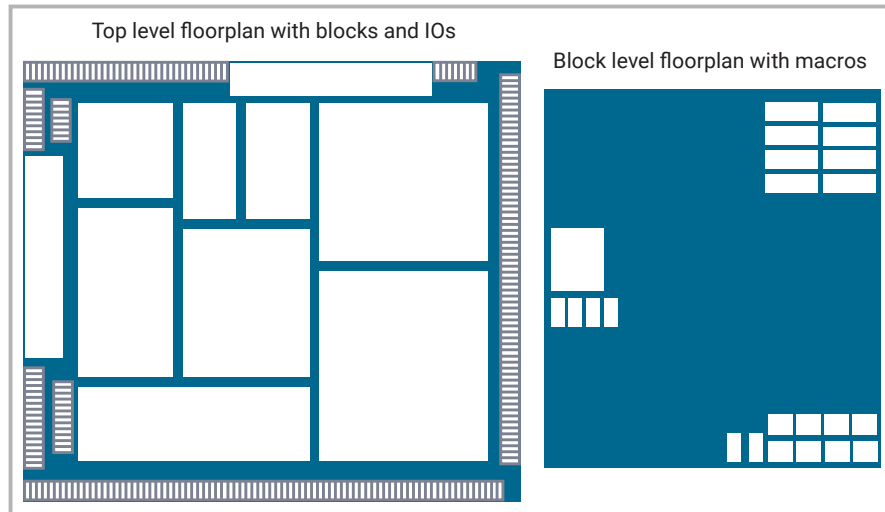


Figure 2: Top and Block Level Floorplans

During the process of exploration it is not uncommon and indeed beneficial for teams to identify implementation issues that are best resolved by re-working the RTL description of the design. Examples of this include:

- Adding physical hierarchy at the top of the logical hierarchy to represent sets of logic modules partitioned into blocks
- Identifying logic structures that create difficult or impossible routing congestion, regardless of how they may be placed
- Recommending the insertion of pipeline stages needed for various top-level signals

Effective Exploration Finds the Best Hierarchical Floor Plan Solution Fast

As described above, the ideal solution offers speed and industry leading automation to produce high quality floorplans, irrespective of input data cleanliness or completeness. This should enable things such as only requiring initial, relative placement as opposed to specific locations, sizes and shapes. Fast, high accuracy analysis tools can then provide efficient understanding and insight into what can be improved. Armed with this feedback, designers need to be able to quickly perform focused and directed refinement to produce the highest quality floorplan. With increasing levels of data completeness, the solution should remain bounded and allow convergence to a final set of data for implementation.

Automation aids designers by quickly generating data to analyze. Many aspects of floor plan creation lend themselves to automation including:

- Die creation and IO placement
- Block placement and shaping
- Macro placement
- Full-chip and sub-block power planning
- Physical pin placement
- Timing optimization and budgeting

As discussed, fast, “relevant”-accuracy analysis tools are fundamental to an exploration solution. These analysis tools enable designers to address problems early thus ensuring a more efficient, convergent exploration process. During exploration, designers typically need to assess:

- Physical partitioning candidates: are there better ways to group logical modules into blocks from a physical standpoint? Can blocks from previous designs be combined into a single block and thus reduce resource requirements?
- Physical block placement and shaping: are block utilizations reasonable? Are the timing of critical top-level busses (refer to figure 3) likely to be impacted by distances between blocks?
- Macro placement: does their placement interfere with inter-block connectivity and data-flow? Is the intra and inter macro spacing sufficient to accommodate transit routing?
- Power routing: does routing intersect cleanly with edges of blocks and macros? Are there any IR drop or EM problems?
- Block pin placement: are the majority of simple, two-pin net connections between adjacent blocks aligned? Are feed-throughs (if allowed) distributed efficiently through blocks?
- Timing optimization and block budgets: are there major timing violations at the top-level? Are the budgets generated for the blocks reasonable? What are the sources of any timing problems of bad block budgets?

The list above shows a subset of the many questions that designers must answer during the floorplan exploration process. Without a rich set of analysis tools available, designers are faced with a host of trial-and-error iterations that can significantly impact the overall schedule.

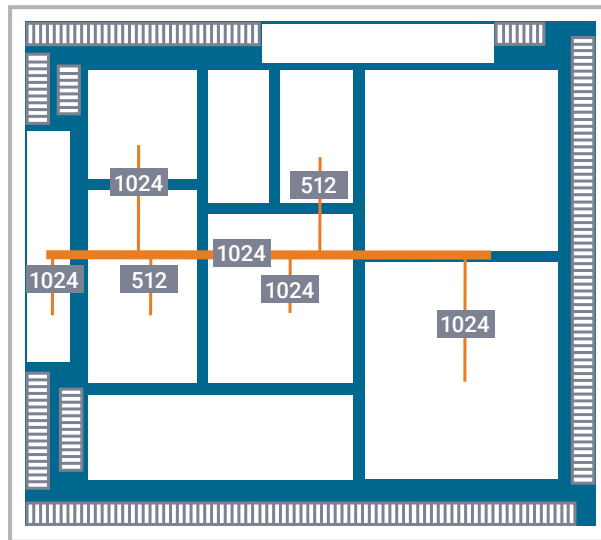


Figure 3: Timing Critical Bus Connecting To Blocks

Based on the knowledge gained from early and then more complete analysis, designers subsequently require tools to refine the floorplan to improve results while minimizing changes to parts of the floorplan that work well. As an example, consider a case where a block size needs to be increased to reduce utilization within the block. If die size must be kept constant, adjacent block areas must be reduced to accommodate the utilization reduction. Interactive editing tools that allow the user to select edges of multiple blocks and move them simultaneously and dynamically see impact on utilization of all of the affected blocks help designers quickly choose the appropriate refinement. Or, consider a case where increasing the block shape requires other blocks to be changed from rectangle shapes to rectilinear shapes. Pins on the formerly rectangle blocks should automatically move to appropriate edges on the new rectilinear shapes.

Intelligent Abstraction Maximizes Exploration Capacity and Throughput

As designers move through floorplanning stages, not all design data is needed. For example:

- Abstracting logic modules as objects with size and module-to-module connectivity provides the information needed to explore partitioning solutions
- Abstracting partitions as blocks with size and block-to-block connectivity provides the information needed to explore block placement and shaping solutions
- Abstracting interface timing segments of blocks provides the information needed to explore top-level timing solutions

An exploration flow that abstracts and reads only data necessary to enable assessment of results of successive floor planning stages produces faster turn times because it requires less memory. It is not necessary to load all design data to complete and assess each floor planning stage.

An efficient exploration solution intelligently creates abstract views that accurately represent the design data needed for a given floor planning stage. Data is added to abstract views as needed for successive floor planning stages. The additional data represents the current floor plan state to provide more accuracy for the next floor planning stage. There is really no good reason for a designer to be forced to read in a full design netlist to create a quality floor plan.

Synopsys' IC Compiler II Provides a Fast, Efficient and Accurate Floorplan Exploration Flow

With the needs of the ideal floorplanning exploration solution in mind, IC Compiler II Design Planning has been written from the ground up for large scale designs. Exploration needs influenced the design of algorithms, data base structure, new analysis tools, and interactive refinement capabilities. The result is a truly state of the art floor plan exploration solution.

The through put speed is unrivaled. Abstraction throughout the floor planning process minimizes memory requirements, further enabling faster through put. Automation to help create a floor plan takes advantage of multi-threading and distributed processing whenever possible.

A rich set of analysis tools, many of them new, helps designers assess results of each step of the process. In addition to helping designers identify problems and find solutions early, the analysis tools provide a natural way of documenting recommended changes to pass back to RTL design teams. New and enhanced GUI capabilities speed the process of focused and directed refinement.

To get started, IC Compiler II reads Verilog and creates an outline view of the design. The outline view contains abstracted views of logic modules that enable designers to quickly and efficiently explore potential partitioning solutions. Only information about module content necessary to accurately represent the modules with size and module-to-module connectivity is maintained. This light weight abstraction of the logic structure enables designers to see and explore the logic structure of the largest of designs in the GUI in real time.

Users see logic modules with size, their hard macro children, and the connectivity between the modules. Designers can:

- See modules sized at a target utilization next to the die outline providing immediate visual feedback in terms of area. Designers instantly see if the logic module hierarchy is too big to fit within the desired die size.
- Explore levels of logic module hierarchy by expanding modules to see their children. Designers can quickly find sets of modules they want to consider as potential partitions. Expanded modules can be collapsed back into their parents if module children are too small to be useful as potential partitions.
- See macro children of modules. Modules with a high percentage of macros vs standard cells generally require more effort for block level floor planning as the macros constrain possible aspect ratios of block shapes.
- Automatically place and shape logic modules. Seeing how modules are arranged within the core helps guide partitioning decisions. For example, placed modules that overlap may indicate they should be grouped into a partition or be expanded into smaller modules to be added to other partitions.
- Group and ungroup modules to perform what-if analysis of potential partitioning solutions.
- Assess partition-to-partition connectivity using data flow flyline analysis. Filtering to focus on critical functional bus connectivity lets designers quickly identify potential top-level timing challenges.

It is important to note that IC Compiler II can handle larger blocks than most tools. Designers can create larger partitions, or fewer blocks, therefore enabling implementation by smaller, leaner designer teams.

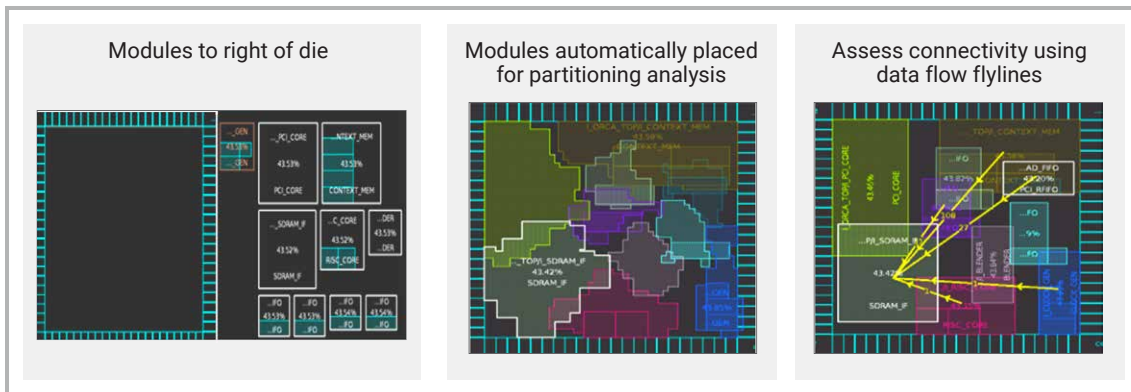


Figure 4: Exploration of Partitioning Using Logic Modules

With partitions defined, IC Compiler II converts the partitions into blocks. Designers provide estimated sizes for black boxes which exist as partitions or within partitions and can guide block placement and shaping by defining:

- Target utilizations on all and individual blocks
- Aspect ratio or specific width and height constraints
- Relative block placement constraints. For example, block A must be placed above block B
- Rectangular or rectilinear shaping allowed on all and individual blocks
- Channeled, with channel size constraints, or abutted
- Block pre-placements within the die

Using distributed processing, abstract views are created for the blocks combining information from the design outline view, block netlists if they exist, and user defined constraints to enable block placement and shaping.

The GUI displays results with block utilizations to aid speedy analysis of results. Designers instantly see if target utilizations are achieved. Net connectivity and data flow fly line analysis let designers see and analyze inter-block connectivity. Net connectivity fly lines show direct connections including connections traced through buffers and inverters. Data flow fly lines enables designers to trace connectivity through user defined levels of logic and registers. By filtering connectivity between blocks, designers can focus on connectivity of timing critical busses. For channeled designs, IC Compiler II provides channel capacity vs. demand and super-fast, what-if congestion analysis maps of channels.

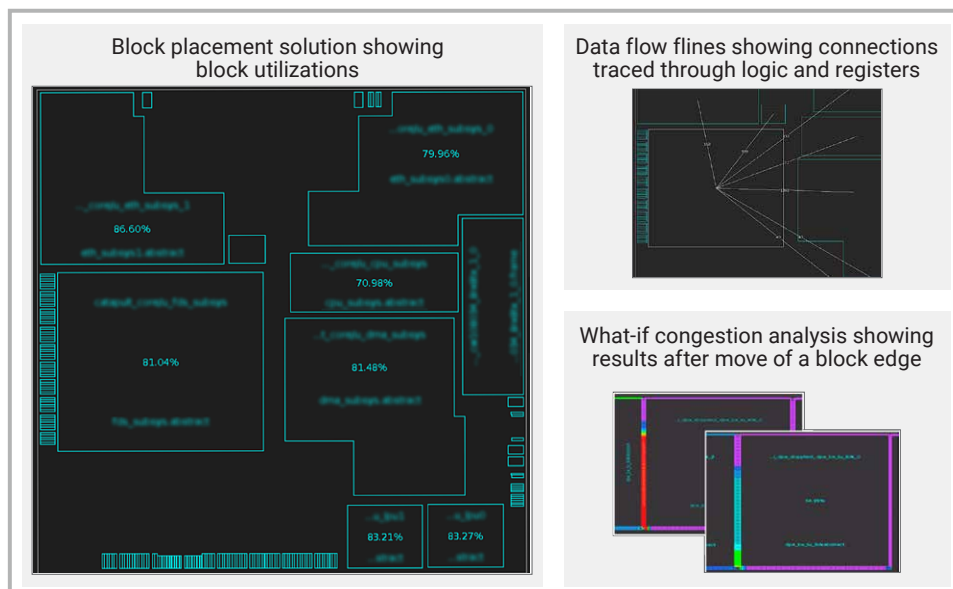


Figure 5: Examples of Block Place and Shape Results Analysis

Armed with information provided by analysis, designers may choose to adjust block shapes and channel sizes. A rich set of editing capabilities enables fast and easy adjustments. Multiple object edges can be selected and moved at the same time. Moves can be snapped to inferred snap points derived from edges of other objects.

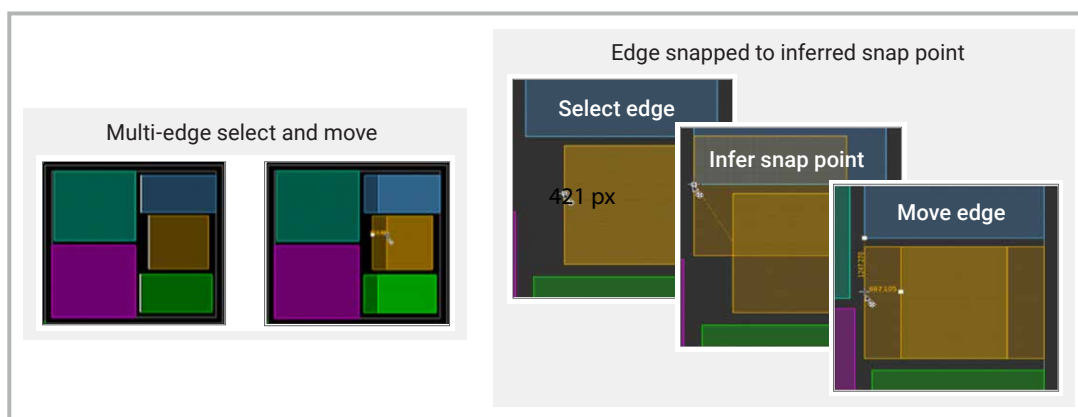


Figure 6: GUI Eases Floorplan Editing

For critical busses, designers can draw corridors to direct their paths between blocks. Corridors can be drawn for the entire path or segments of the path. In most tools, designers would start measuring distance in microns between blocks or along bus corridors to help them calculate time-of-flight delays using RC values for metal layers pulled from tech files. IC Compiler II has a patented timing ruler that allows designers to define layers and buffers to use to draw rulers.

The ruler presents units of time, rather than distance, making it much easier for designers to see time-of-flight delays. The need for pipelining stages can be quickly assessed, documented, and fed back to RTL teams.

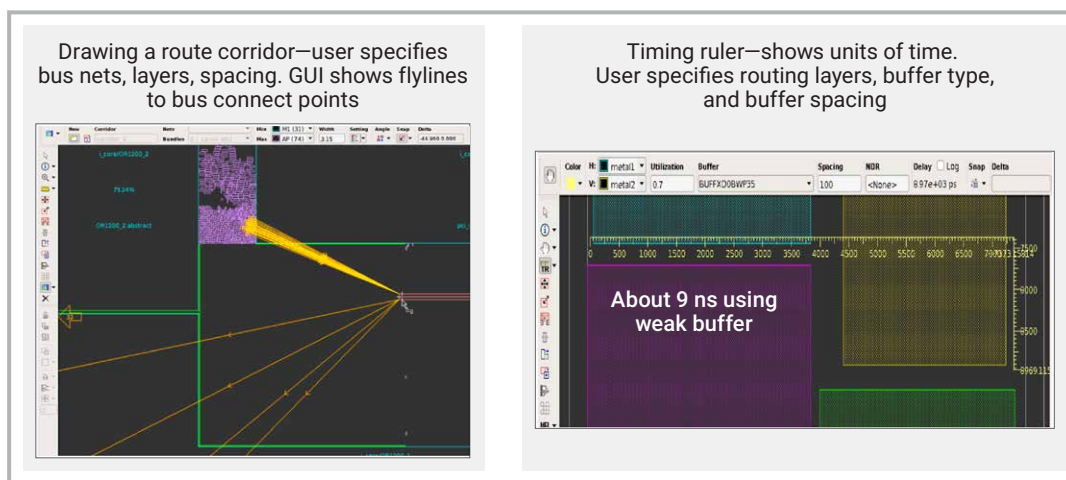


Figure 7: Route Corridors and Timing Ruler Help Users Plan and Assess Critical Busses

IC Compiler II supports global cell placement for the entire design. Using distributed processing techniques the placement engine considers block shapes, inter-block connectivity, routing corridors, block-pin constraints, and data flow to perform global placement of cells. Block abstracts are automatically updated to include placement information which helps analysis of results and later guides hierarchical pin placement.

With the updated abstracts, designers can more accurately analyze the impact of placement upon inter-block connectivity. Additionally, the global placement creates macro placement within blocks based on full-chip data flow providing block designers with a better starting point for their floor planning tasks. In many flows, block designers are tasked with creating their own macro placement using hierarchical block pin placement taken from the top-level floorplan.

IC Compiler II's GUI provides multi-level hierarchy viewing that enables designers to quickly see results of global placement. Net connectivity and data flow flyline analysis can be re-applied to assess results. Given the global macro placements, data flow flyline tracing can now include paths to and from macros placed within blocks. Global routing sees more accurate connect points at all levels of hierarchy to produce congestion maps.

If designers want to manually modify results, the GUI supports transparent hierarchy editing. Objects from any level of hierarchy can be selected and moved while seeing the full chip context. Top-level designers can make global refinements without having to open additional sessions to edit block content.

IC Compiler II creates power routing by applying power routing strategies to user defined power regions. Regions are defined in terms of floor plan and design objects, such as voltage areas, groups of macros, etc. Re-working power routing required by floor plan changes is eased because regions can be re-derived and strategies simply re-applied.

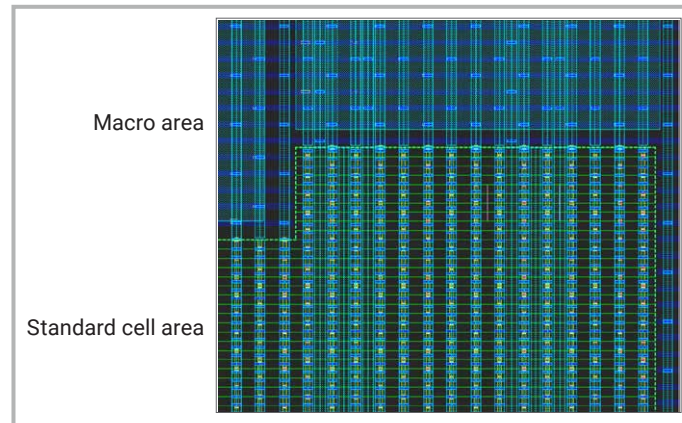


Figure 8: Power Routing Created Using Strategies Applied To Power Regions

The pin placement engine considers block placement, cell placement, routing corridors, power routing, and uses the global routing engine to determine where best to place hierarchical pins of blocks. The engine supports channeled or abutted styles. For channeled styles, pin placements of nets connecting adjacent blocks are aligned to enable straight wire connections across channels. This insures routing resource in channels is not consumed by jogs to connect un-aligned pins. For abutted styles, pins of adjacent blocks are aligned and the pin placement engine adds feedthroughs to blocks to form connections of nets between blocks that do not have common edges. Designers can further control pin placement using:

- Pin constraints that are global, by nets, by busses, and by block references. This includes allowed layers, spacing, side, order...
- Feedthrough controls, which are global, by nets, by busses, and by block references. This includes allowed or not allowed and specifications of desired feedthrough paths through blocks
- Pin guides and blockages

Designers can guide pin placement based on experience and knowledge of the design gained through analysis. It is worthy to note that global placement considers pin placement control that is defined before placement.

Data flow flyline and routing congestion analysis can be re-applied to help assess QoR. Additionally, a pin checking utility shows pin alignment and feedthroughs. Designers can quickly assess and document results.

Summary

Today's designs are huge and very complex, requiring hierarchical planning and implementation methodologies. An efficient floor plan exploration is required to:

- Produce a quality floor plan fast
- Provide constructive feedback and recommendations to RTL design teams
- Create block level physical and timing constraints
- Reliably predict the feasibility of achieving the target area, timing, and power specifications

A fast, accurate solution enables design teams to assess more solutions and converge on the best solution more quickly within the time scheduled for floor plan exploration. IC Compiler II provides solution that enables designers to find the best floor plan for implementation, fast.