WHITE PAPER

DESIGN WITH PHYSICAL: BRINGING PREDICTABILITY TO LOGIC DESIGN

cadence™

BACKGROUND

For some time now, wireload models have been inadequate for modeling wire delay accurately, and the problem is getting worse with each new process generation. The consequence is that logic designers see a different timing representation of their design than what they see in physical design. This discontinuity affects the ultimate success of the project in several ways. For instance, if the wireload models under-represent actual physical wire delay, then logic designers will see a more optimistic timing representation than what appears in physical design. This typically leads to long iterations between the logical and physical design teams, as each team sees a different timing representation and makes optimizations based on disparate assumptions.

Another result of this discontinuity is that the logic design team might build in margin by using wireload models that over-represent real physical wire delay—a pessimistic timing representation—in which case the design will use larger and higher power cells even for non–timing-critical logic. As a result, the chip will be bigger or more congested, and it will consume more power than necessary. This tradeoff is no longer acceptable in today's market, where much growth comes from high-volume, low-margin consumer products for which power dissipation is a key concern. As such, it is now essential to employ a design methodology that more effectively captures physical effects during logic design and implementation.

CURRENT SITUATION

Today's complex nanometer designs undergo a combination of these effects, offering the worst of both worlds: even a pessimistic wireload model will underestimate some of the long routes in the physical world. The outcome is that these long routes still are under-powered and will cause timing closure iterations, while the rest of the logic is over-driven and will consume too much area and power. The underlying problem is that the logic design team creates and hands off gates without any insight into physical wire effects. The real measure of a design's integrity is "quality of silicon" (QoS), which is timing, area, and power measured with wires. However, this measurement cannot happen until some physical implementation is complete. Figure 1 highlights the gap between what the logic design team creates and what the physical design team sees.

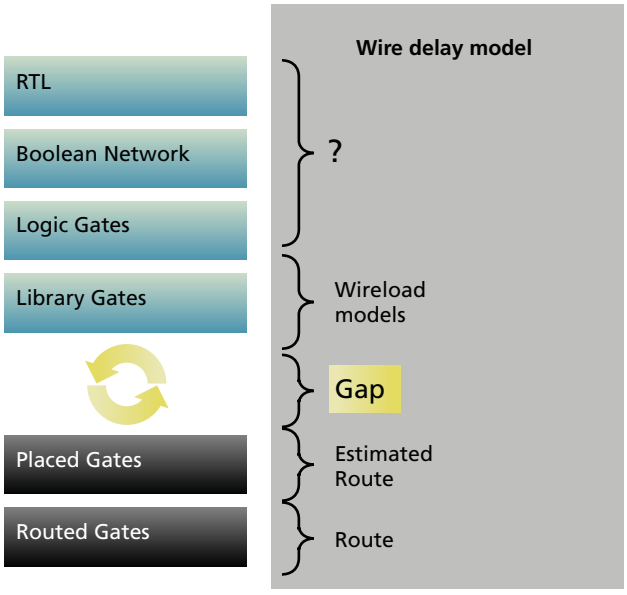

Figure 1: The logical-physical wire-delay modeling gap

Physical synthesis was developed to address this problem by using placement and some form of routing to achieve more accurate wire delay information. However, physical synthesis is also bogged down by all the detailed data associated with it, and is constrained by the initial netlist and placement. The result is low capacity, long runtimes, and only local incremental optimization capability. This works well in physical design, where the detailed accuracy is necessary—physical synthesis provides more optimization power than was previously available here. But in logic design, all these details are not yet needed—it's only important that you have enough accuracy to know whether you're moving uphill or downhill on the cost function.

In addition, design-for-test (DFT) logic can affect and be affected by physical implementation. Scan-chain connection that is ignorant of register placement can result in both setup and hold timing violations in physical design, not to mention excessive amounts of routing overhead. And DFT logic such as compression, BIST, and boundary scan need to be placed near I/Os and macros, often causing congestion and blockages that add to the timing closure challenge.

As a result, design teams need to do the best job they can at modeling physical effects at every level of the design process without sacrificing optimization capability. For logic designers, wireload models enable full synthesis optimization capability, but are inadequate for modeling timing. Physical synthesis provides more accuracy than is needed at the logic level, and therefore does not offer the necessary optimization capability. This is the source of the disconnect between logical and physical design. However, with physical wire effects dominating the results, what can be improved to improve physical timing accuracy while still providing the optimization capabilities required by logic designers?

1. **Better physical modeling in RTL-to-gates synthesis.** Physical synthesis tools may claim to synthesize "RTL to placed gates," implying placement-aware RTL synthesis. However, placement cannot be performed until gates are available—and gates are not available until RTL-to-gates synthesis is performed. What do engineers use to model wire delay during the RTL-to-gates step? Either inaccurate wireload models or, in the worse case, zero wireloads. Physical synthesis tools claim that RTL-to-gates synthesis is not important to quality of silicon (QoS). However, new global synthesis has proven that RTL-to-gates synthesis is key in improving chip frequency, area, and power. So how can design tools provide a real-enough physical timing representation to the RTL-to-gates synthesis process?
2. **Full synthesis optimization, starting from RTL.** Physical synthesis greatly increases the optimization capability of physical design. From the logic designer's viewpoint, physical synthesis does less optimization than logic synthesis because it starts off constrained by a placed implementation. During logic design, optimization decisions are at a higher level: what kind of adder architecture to use, whether to employ resource sharing, and so on. These optimization decisions cannot be performed once the design is at the netlist or placement level. It would help to have some level of physical timing reality as input, but this level of optimization does not require the type of detail associated with placement and routing.
3. **Effective logic-design–oriented use model.** Physical synthesis tools require too much physical knowledge for most logic designers to effectively run and analyze their results. Yet physical synthesis of a netlist, constraints, and a physical library without a floorplan or other physical parameters can produce less than optimal results. SoC designs today have hundreds if not thousands of macros, use multiple power supplies and complex I/O schemes, and present many other challenges that need to be considered during implementation. And if the design does not meet timing after running physical synthesis, then what? All analysis in physical synthesis is gate level or physical. The ideal solution is to provide realistic physical timing information while maintaining a use model in which logic designers can still be productive.

While there is not a *single magic bullet* that can address all of these issues, there are new methodologies that can be employed at each step of the design process to bring physical timing into the logic design world in an effective manner.

DESIGN WITH PHYSICAL METHODOLOGY

RTL-TO-GATES SYNTHESIS

This is the most important implementation step in terms of optimizing your design goals—timing, area, and power. Where the global logic structure is created is the starting point for all incremental optimizations that will follow. Until gates are created however, there is no way of knowing actual placement and wire delay. But there are several ways we can model this that are significantly better than conventional static, fanout-based wireload models.

It is well known that using fanout as the only factor in wire delay is the most prevalent shortcoming of wireload models. Here are several issues with traditional wireload models.

1. Non-monotonic behavior: Wireload models are lookup tables. If there are not enough data points, the result is often non-monotonic load values, delivering unpredictable results out of synthesis.
2. Pessimism: It is natural to build in extra timing margin by using conservative wireload models. However, this conservatism affects all logic, not just that which is timing critical. The result of this pessimism will be higher area and power consumption than necessary for much of the design. And even a pessimistic wireload model will underestimate the delay of the small percentage of long wires in the design.
3. Inability to adapt to change: A custom wireload model from placement is obsolete after the first optimization because the design has changed. The realistic case is that the design evolved—both RTL and constraints—while the physical design team was generating the placement and wireload models.
4. Inability to model different classes of designs: A single lookup table must be used for each synthesis run, but synthesis can be run on two million or more gates at a time with modern synthesis tools. Within all that logic exists numerous types of logic structures with various routing characteristics. Consider the difference in routing characteristics between a large mux and a datapath segment. The mux will have many long wires connected to it from all over the place, while the datapath will contain primarily short local routes through it. A single wireload model cannot capture this.
5. Coarse granularity: A library will only have wireload tables for certain design sizes (100k, 500k, and so on). But what if your design falls in between? Designers must choose between being aggressive or conservative, often having to experiment to find the one that gives the best results out of physical design, which is a time-consuming process.

The inadequacy of wireload models has driven some designers to synthesize without wireload models at all, leaving the entire problem to physical design. This technique would be okay if the synthesis tool only performs very simple timing optimizations such as buffering or upsizing, which can be done more accurately in physical design. But if the synthesis tool performs more sophisticated global logic optimization for timing, area, and power simultaneously, then wire timing cannot be ignored. Designers need a better technique to provide more accurate timing information to RTL-to-gates synthesis.

This technique needs to capture physical implementation tool behavior during RTL-to-gates synthesis. Whereas wireload models use a static representation of area combined with a fixed-entry lookup table based on fanout, modern synthesis tools should be able to calculate area and fanout dynamically and combine them with physical library information for more accurate modeling.

Having a more realistic timing model for the design would remove potential bias produced by overly optimistic or pessimistic wire delays, and it would eliminate the need to build in an extra timing margin at the expense of area and power. A more realistic timing model would need to replace wireload models with physical library delay information, while being simple to adopt and without runtime penalty. This would result in better QoS—timing, area, and power measured with wires—because it more accurately directs synthesis optimization to apply the proper types of optimization to each area of the design. Most importantly, a more realistic timing model would achieve the best possible results while eliminating the need to experiment with different wireload models.

NETLIST-BASED PHYSICAL PREDICTION

Using the recommendations outlined earlier would enable synthesis to better model the local wires in a design, which typically cover 80-90% of all wires. The other 10%, however, is what often causes the biggest problems in logical-physical closure. There is only one way to identify long wire issues accurately, and that is to perform production placement and routing using a real floorplan.

Silicon virtual prototyping (SVP) was designed for this. It has proven to be a fast and high-capacity method for generating an accurate physical view of the chip. Any other technique is misleading because it will result in "false" long wires. However, for most logic designers, SVP requires too much physical information to get started, and even if you do get it to run and generate an accurate view of your physical timing issues, then what do you do?

The solution is to link synthesis with silicon virtual prototyping. Modern synthesis tools can write-out setup files to drive SVP, so the next logical step is to encapsulate the SVP run in a single command from the synthesis cockpit. The physical design team would provide the production floorplan, although the solution should also allow the RTL design team to generate a floorplan automatically before the production floorplan is available. In fact, such a solution would enable the two teams to collaborate on the floorplan.

But what happens once the physical prototype is created? The results should then be brought back into the synthesis environment, where the synthesis user can analyze physical-based timing in a familiar environment and take corrective action as necessary:

- If timing, power, or area is not close to being met, then the designers are still in the synthesis environment. This means they should have full ability to perform more synthesis optimization, adjust constraints, or take other appropriate action. Then they could re-run prototyping for another update of physical prediction.
- If the design goals are met, or are close to being met, the logic design team should be able to export the results to the physical design team: the netlist, constraints, as well as the placement used to signoff on timing, power, and area. Passing forward this placement ensures that the physical design team starts with the same view of the design's timing, power, and area characteristics seen at handoff, thus eliminating surprises.

In short, a solution that embeds silicon virtual prototyping within synthesis should deliver production-accurate physical timing information into the logic synthesis environment. This would allow the RTL team to own the timing closure process before handoff to physical implementation, and would eliminate the risks associated with this handoff.

CONCLUSION

While there is no single, simple use model to solve the complexities of wire delay modeling and timing closure, this paper proposes a two-step approach to improving the process substantially:

1. Improve wire delay modeling in RTL-to-gates synthesis using physical library information and dynamic design modeling. This is a simple-to-adopt solution to a long-standing problem. While it does not effectively address the long wires in the design, it does effectively address the other 80-90% of the wires.
2. Address the remaining 10% of wires (that are long and problematic) by using production physical engines with the production floorplan as an input. The key is to make the use model synthesis-oriented. This proposed solution would invoke silicon virtual prototyping automatically and annotate the results back into the design in the synthesis environment, enabling logic-oriented analysis and synthesis re-optimization.

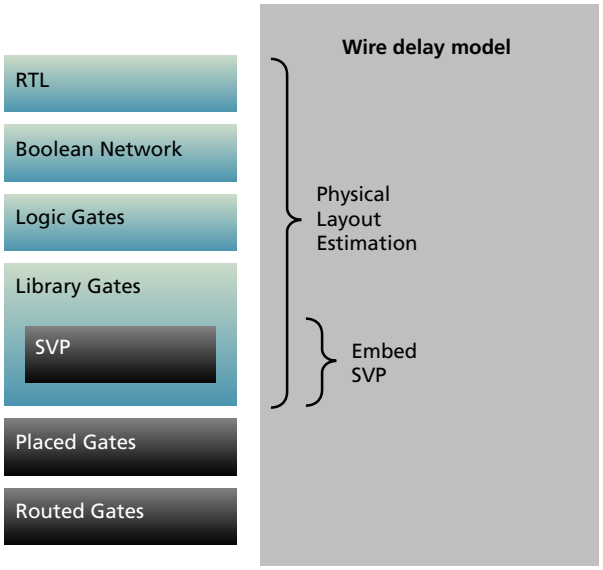

Figure 2: A pragmatic solution to deliver accuracy-appropriate physical wire delay information to synthesis

This two-step approach would allow RTL designers to perform the job they are accustomed to, while utilizing better physical timing information than was previously available. Most importantly, this approach bridges the gap between synthesis and physical implementation, providing the logic design team with the ability to own design closure and the physical design team with a better starting point for implementation.

cadence™

Cadence Design Systems, Inc.

Corporate Headquarters
2655 Seely Avenue San Jose, CA 95134
800.746.6223 / 408.943.1234
www.cadence.com

7006C 05/07 MK/GDA/JA/PDF