

# SAME 2002

## Session: 2

### RAPID HIGH CAPACITY PROTOTYPING AND PHYSICAL SYNTHESIS

Olivier Coudert, Monterey Design Systems

Sunnyvale, CA, USA

#### Abstract

Deep-submicron (DSM) processes have made impossible to sign off a gate-level netlist with full confidence that the final physical implementation will meet all design requirements. Since interconnect has a significant impact on timing closure, and signal integrity issues cannot be handled without layout specific information, design signoff can no longer be accomplished solely by logic synthesis. Because of this, two new concepts have arisen: physical synthesis, and physical prototyping. Physical synthesis is logic optimization together with place-and-route. We define physical prototype as a circuit description with enough physical information to deliver a design signoff. Section 1 will discuss the design closure aspects, and Section 2 will discuss physical prototyping.

Another challenging aspect of today's designs is their ever-increasing design size and complexity. This can be addressed with a hierarchical methodology. Section 3 will explain how physical prototyping can be used to enable a top-down planning, bottom-up implementation, hierarchical flow, which restores the ability to sign off designs prior to the completion of the final physical implementation, and which allows a continuous validation of the chip-level constraints throughout the design process.

#### 1. The Impact of Deep Submicron on Physical Design

Since the advent of logic synthesis in the mid-1980's, design flows have been characterized by a clear separation between the logical and the physical domains. The point of delineation was known as 'sign off'. Logic synthesis tools would produce a gate-level netlist from a high-level specification, usually RTL. Place and route tools then transformed the gate-level netlist into a physical layout. Place and route tools were not allowed to change the netlist because this would have invalidated the signoff process. However,

there was really no need to change the netlist because cell delays dominated the timing of the chip and the impact of the physical layout on timing was small.

#### 1.1 Timing Closure

The gate delay depends mostly on the output capacitance it drives, of which the net capacitance becomes the largest contributor; also the delay of long nets, which depends on their capacitance and resistance, becomes larger than gate delays. At 0.25 $\mu$ m, interconnect becomes the dominant factor in determining delay, so we can no longer ignore the effects of physical implementation on timing closure, and the ability to sign off a gate-level netlist is lost. Legacy design flows that use statistical wire load models (WLM) to estimate the interconnect delays are no longer effective, because timing is determined by the maximum path delay: although a WLM may be a good predictor of the average wire load, the deviation is so large that timing ends up to be grossly mispredicted.

Moreover, coupling capacitance (the capacitance between nets on the same layer) becomes more dominant over inter-layer capacitance (the capacitance due to overlapping of interconnect between different layers) with every new process technology. This is because the nets are getting much closer to each other and the wire aspect ratio of height to width is increasing. In 2002, the coupling capacitance is more than four times larger than the inter-layer capacitance, and the ratio is projected to increase to six by 2010. Thus the capacitance of a net cannot be determined without knowing both its route *and* that of its neighbors.

#### 1.2 Signal Integrity

Finer geometry, higher clock frequency, and lower voltage produce signal integrity problems. For instance, since the inter-wire coupling capacitance dominates the inter-layer capacitance, crosstalk can no longer be neglected, because it has a primary effect on timing, and it induces signal noise. IR-drop's impact on timing is becoming more visible

with lower voltage. Other physical aspects such as antenna effect, electromigration, and self-heating, need to be analyzed and controlled. At 65 nanometers and below, implementation tools will need to consider inductance. Signal integrity problems must be addressed as early as possible during implementation since it is very costly and time consuming, if not impossible, to fix them by post-processing a completed layout.

## **2. Physical Synthesis and Prototyping**

In the mid-1990's, the first generation of physical synthesis tools appeared and consisted of crude attempts to combine different point tools performing logic optimization and placement within an integrated flow [7]. The current generation of physical synthesis tools successfully achieves this goal. However, at 0.13 $\mu$ m and below, physical synthesis alone is not sufficient to restore the ability to sign off a design prior to the completion of the layout. For this, physical prototyping is required.

### ***2.1 Physical Synthesis***

Physical synthesis addresses many DSM issues by combining elements of logic synthesis and physical implementation together into a single tool. Physical synthesis, as most people use it today, starts with a gate-level netlist and performs logic optimization, placement, and global routing, to produce a placed design that meets timing requirements. Physical synthesis may employ numerous techniques to optimize the logical structure of the chip including: gate sizing, buffering, pin swapping, gate cloning, useful skew, re-synthesis and technology re-mapping, redundancy-based optimization, and area and power recovery.

This is a significant improvement over pure logic synthesis because the logic optimization is performed and evaluated based on cell placement that reflects the final placement. The output of physical synthesis is passed along to an implementation tool that completes the job by adding power and clock networks, adjusting the placement as needed, and then by detail routing the design. After the routing is complete, the design may be checked for signal integrity issues.

It is significant to note that it no longer makes sense for RTL-to-gate synthesis tools to perform sophisticated gate-level optimization. Without accurate physical information, logic synthesis tools cannot make good decisions about cell sizing or buffering. Physical synthesis is much better suited for these tasks. Today, the role of RTL-to-gate logic synthesis has been reduced to simply produce a gate-level netlist with the adequate structure as quickly as possible, and then pass it along to

physical synthesis without attempting to optimize the sizing or buffering aspects.

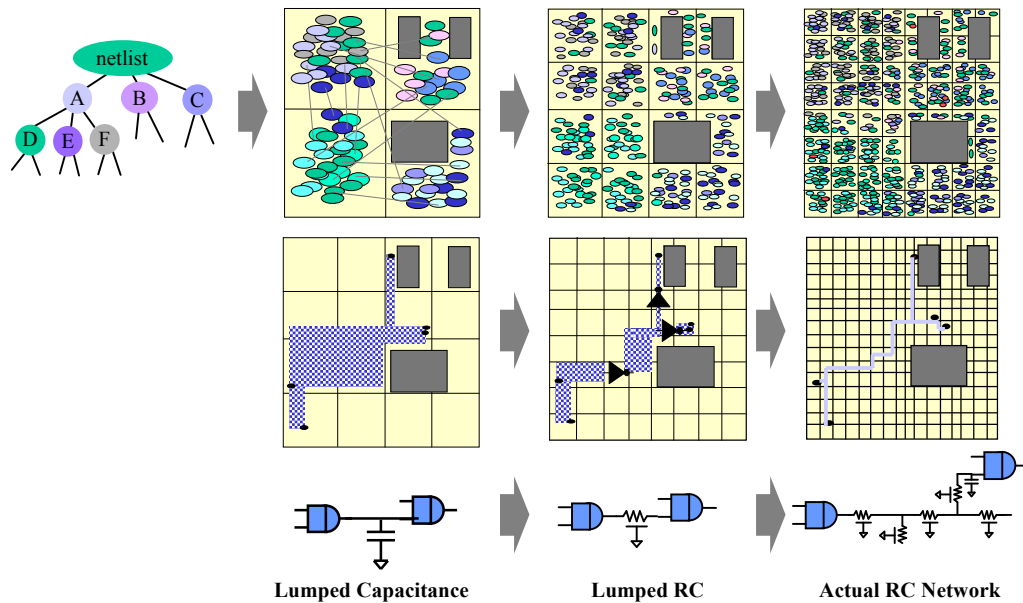
### ***2.1 Physical Prototyping***

As processes progress below 0.13 microns, it is becoming apparent that physical synthesis alone is not enough to predict the final physical implementation. The primary shortcomings are the omission of accurately modeled power and clock networks, scan chains, and signal integrity issues. Power and clock networks cause massive perturbations upon the cell placement and consume large amounts of routing resources; they cannot be ignored or abstracted during physical synthesis with the expectation that the resultant placement or global routing will correlate closely to the final layout. The construction of a physical prototype not only includes logic optimization, placement, and global routing. It also includes fully routed power networks, fully synthesized and placed clock trees, and a global router that allocate enough routing resources in problem areas so that the detailed router can later avoid the potential signal integrity violations. A physical prototype is a ready-to-route implementation that correlates to within a few percentage points of the final, detail routed design.

### ***2.2 Design Closure With Continuous Refinement***

Although physical design is often presented as a timing closure problem, the reality is that all design variables need to be considered together. Besides dealing with the traditional timing/area/power triangle, we also need to address congestion, clock tree synthesis, scan-chain reordering, and signal integrity. The increasing dependency of the design quality on physical effects must be managed.

Some of the proposed solutions to solve the DSM challenges have been discussed in detail elsewhere [2]. One approach to deal with these complex inter-dependencies is to start with a very early estimate of the physical implementation using a very coarse level of granularity, and then to continuously refine it at progressively finer levels of granularity until the final implementation is complete. This approach, combined with an open, multi-variable cost function, exhibits a number of very desirable attributes. 1) It enable simultaneous optimization of all design requirements, 2) it provides continuous feedback on the progress towards attainment of the design goals, and 3) it can be paused when a pre-determined level of granularity is achieved and signed off prior to final physical implementation.



Prior to the completion of the physical implementation, design aspects may be estimated, but always with some degree of uncertainty. For example, the actual timing of a design can only be calculated if all routes are fully known. If the design has only been partially placed and routed, then the uncertainty on net capacitances results in an even larger uncertainty on timing. Experience shows that until sufficient information about the placement is available, it is not possible to optimize for timing beyond simple wirelength minimization.

If we know the precision with which placement and routing are determined, we can determine the parameters of the design that can be estimated with enough accuracy to allow some valuable optimization. As the precision of the placement and routing is increased, new design parameters become "visible" and can be optimized in turn. A refinement-based flow builds the physical design step by step, by increasing the resolution of every parameter simultaneously. At the beginning of this process, there is only an approximate placement and global routing, and the clock tree and power/ground network are roughly estimated (their contributions to the congestion must be accounted for as early as possible). At the end, there is a fully detailed, placed and routed netlist, including the completed clock tree and power/ground routing. Between these two points, there is a continuous progression from rough to detailed, and all design variables are monitored and optimized simultaneously (when their resolution allows it) to meet the design constraints. As the design implementation progresses, the models become more accurate (the estimations have less uncertainty), and the actions taken to solve the problems are more and more detailed and localized. This continuous process of model refinement and design variable monitoring and optimization is the key to prediction and convergence.

This refinement process is illustrated in the figure above: placement of smaller and smaller clusters of cells is done in smaller and smaller bins (see SAME 2002, October 10<sup>th</sup> 2002

Section 2.2.1); global routing goes from describing the area a net is likely to span, to a reduced route after logic optimization seeded the route by dropping buffers, to a nearly finalized route. Timing models can range from lumped capacitance at the beginning, to lumped RC when more placement information is available, and finish with a full RC network.

At a certain point in the refinement process, far in advance of the final physical implementation, the design will have stabilized sufficiently so that timing can be predicted within 5-10% of the post-layout clock cycle. It is at this point that the design qualifies as a physical prototype and can be signed off.

### 2.2.1 Placement

A variety of methods have been applied to the placement problem [2]. Multilevel hypergraph partitioning [3] speeds placement by allowing groups of cells to be moved together. This cluster move-based method with an open cost function allows simultaneous optimization of all design aspects (including timing, congestion, wirelength), which makes it a good candidate for a refinement-based flow. At the beginning of this process, we have a rough placement in a few bins. At the end, we have an accurate placement with a few dozen cells per bin. Even there, the placement is still flexible (i.e., cells have no precise  $xy$  coordinate yet).

This flexibility of the placement is essential to the notion of simultaneous optimization because it allows for netlist changes throughout the entire process until the final placement and routing are complete. For example, as logic optimization is performed concurrently with placement and routing, changes to the netlist are easily absorbed even very late in the process [16, 5]. This approach is also very receptive to incremental engineering change orders (ECO's). Changes to the netlist may be localized to a small number of bins at some level

of granularity and the perturbations may thus be limited to only those bins.

### **2.2.2 Logic Optimization**

The gate-level netlist is initially synthesized with no placement information and, therefore, only with a crude estimation of delays – unless one uses constant-delay synthesis [6] with a high degree of confidence. Timing becomes meaningful when enough placement information is available. Only at that moment can the logic be revisited with a good understanding of timing, as well as congestion and power. Physical synthesis combines logic optimization, placement, and global routing into a single procedure. It changes the netlist to optimize physical aspects of a design (timing, area, power, congestion, etc) in the context of place and route information. It is significantly different from gate synthesis, since it has many more parameters to consider, and must interact with the placer and router.

Common physical logic synthesis and optimization techniques are: (1) load and driver strength optimization such as gate sizing, buffering, pin swapping, and gate cloning; (2) timing boundary shifting, including transparent latches (i.e., cycle stealing), useful skew, and re-timing; (3) re-synthesis and technology re-mapping; (4) redundancy-based optimization; and (5) area and/or power recovery (e.g., low voltage threshold cells have a high leakage power, thus power recovery is becoming crucial for high performance design with dual Vt). Some new logic transformations specific to physical synthesis, such as synthesis of both the logic and interconnect [4,5,8], synthesis for congestion [1] or logic optimization for signal integrity, are also emerging.

### **2.2.3 Clock Tree Synthesis**

In traditional flows, clock tree synthesis is a separate task. Typically, it is built after the placement of the sequential elements and gated clocks has been finalized. However, synthesizing clock trees after placement creates routing problems that may only be resolved by changes to the placement, which in turn disrupts the timing. Clock tree synthesis must be part of the refinement process, because it significantly affects routability and power dissipation, as well as the timing because of skew. Note that the once-desirable goal of achieving a zero-skew clock tree has two important drawbacks. First, non-zero skew can be used to optimize timing. Second, a zero-skew clock tree implies that all the sequential elements will switch at the same time, which may produce an undesirable power spike.

At some point in the refinement process, the distribution of the sequential elements and gated clocks will stabilize to the point where any further changes, bounded by a few bins, will be within an acceptable tolerance. At that level one can determine the amount of routing and buffers needed to carry the clock signal, and the trunk of the clock tree can be built.

### **2.2.4 Power/Ground Routing**

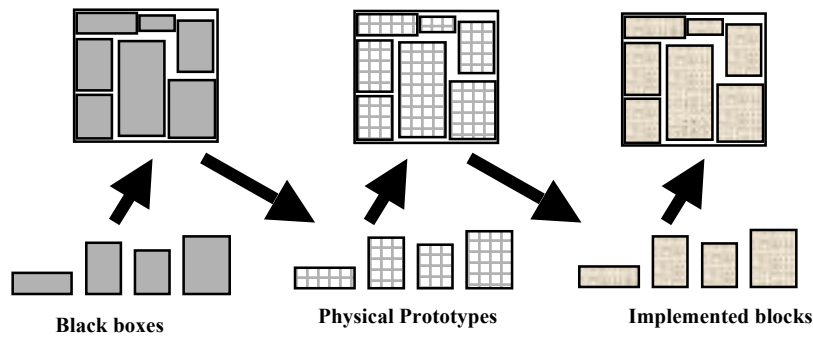
The power network has a major impact on routability. For that reason, it must be constructed as early as possible so that its effect on routability may be accounted for during the entire implementation process. Initially, power routing is performed according to a user-defined routing topology that may include special structures such as chip-level rings, block-level rings, and power grids. At some level of refinement, IR drop analysis can be performed to check the reliability of the power network. This allows an accurate assessment of the quality and integrity of individual power wires and adjustments may be made accordingly.

### **2.2.5 Routing**

Once the physical prototype has been signed off, the refinement process can be resumed until detailed placement, and a detailed router is needed to complete the layout. The detailed router should support both gridded (for speed) and gridless (for detailed optimization) modes. It must support variable wire width and spacing. It must enforce numerous DSM routing rules such as metal/via antenna, end-of-line, minimum area, via array generation for wide-wire connections, width-dependent spacing. Also, it must have timing, congestion, crosstalk, and electromigration awareness at all times.

## **3. Hierarchical Design & Physical Prototype**

Hierarchical design is a necessary direction to handle the capacity and project management problems of multi-million gate designs. This section discusses the use of physical prototyping to enable a predictable DSM hierarchical flow.



There are several reasons to use a hierarchical flow. Designs with 10+ millions gates require partitioning to overcome the capacity limit of tools. Note that the tool with the least capacity (e.g., parasitic extraction, physical verification) will become the limiting factor for the complete flow. Note also that a hierarchical methodology may be the only way to achieve functional verification. A hierarchical methodology allows multiple teams to work on different parts of the design concurrently and independently. This “divide and conquer” reduces the complexity of the design problem for each design team, and reduces the time to market. It also allows for faster ECO cycle times, as functional changes may only affect a local block and thus the remainder of the design is not affected.

A typical SoC flow consists of design planning and budgeting, block implementation, and chip assembly. Design planning is about determining the chip-level constraints. Budgeting consists of determining the block-level constraints that will guarantee, if met, the feasibility of the chip-level constraints. Planning and budgeting includes area and timing budgeting, top-level routing and buffering, power and clock distribution. The blocks are then implemented within their allocated budget, and finally assembled together. Once the top-level design plan has stabilized, the blocks can be allocated to multiple, independent, teams.

### 3.1 Top-Down Budgeting, Bottom-Up Implementation

A pure top-down budgeting approach relies on the assumption that the budgeting does not need to be revisited. Unless being very conservative, it is impossible to predict upfront whether the block constraints can be met. Also it is difficult to adjust the budgeting if we cannot capture the physical properties (e.g., driver strength, parasitics, current drain, etc) that are observed at the block and chip boundaries. This section describes a top-down budgeting, bottom-up implementation, hierarchical flow, which overcomes these limitations with the use of physical prototyping.

In practice, the blocks behavior and physical characteristic are not fully known during the planning phase, thus rough estimates are used instead. As the blocks become progressively better defined, it is necessary to relay the new blocks information back to the chip-level, which is

incrementally updated, and the appropriate adjustments are pushed down again to the blocks level. This leads to a top-down budgeting, bottom-up implementation flow, which is more reactive and more predictable, and which allows chip-level validation of the constraints during the lifespan of the project.

The top-down budgeting, bottom-up implementation, is illustrated in the figure above. Simple block models (here, black boxes) are used to derive a floorplanning and budgeting. We then build the physical prototypes of these blocks within their budgets. These physical prototypes reflect the final implementation of the blocks, but are obtained in a fraction of the time needed to produce a full implementation. The physical prototypes are then re-injected at the top level, so that we can refine the chip-level constraints. The budgeting can be adjusted accordingly, using the physical prototypes of the blocks as accurate models. When the final budgeting is agreed on, we return to the blocks and resume their implementation, and we finish with the chip assembly.

Note that the same principle can be applied with a mix of soft and hard blocks and glue logic. Physical prototypes are accurate physical block abstractions that can be re-injected at the top level at any time, which allows the team responsible for the top level to continuously validate the chip-level constraints, and to propagate the budget changes down to the blocks. Also different level of abstractions can be mixed at the top level, enabling early verification and adjustment of the chip-level constraints while some block are still being specified, some are being implemented, and some are already finished.

### 3.2 Abstraction models

Planning needs timing, area, and power estimates of the blocks. It can be rough estimates, together with an aspect ratio range so that floorplanning has more flexibility. From these estimates, floorplanning, pin assignment, top-level routing & buffering, and channel sizing can be done. Once the planning phase has stabilized, budgeting can be done. This requires the capability to work with models of several level of abstraction. E.g., timing can go from a fixed delay black box, a timing abstraction

where the details of the logic are hidden with timing arcs between I/Os and clock signals, a physical prototype, or a full netlist with parasitics. Timing models and abstractions have been studied and used in the industry for a long time. But new block abstraction formats are needed, e.g., to model how the current is drained from the P/G network, or to model the crosstalk impact of over-the-block routing.

When we want to estimate the chip-level performances, or optimize some glue logic between blocks, we can use the physical prototypes of the blocks to model their physical behavior. When implementing a block, we also need the physical behavior of the chip around the block. Again, timing and power behavior are needed, and can be obtained through physical prototype, but there is currently a lack of a common format to capture the notion of chip context (strength of the drivers of the block, parasitics driven by the block, capacity of the P/G network, skew of the clocks, etc).

### 3.3 The Role of High-Level Synthesis

Section 2 explained why RTL-to-gate synthesis should not try to perform costly gate-level optimization, but only focus on producing a good logic structure –resources sharing, logic depth or gate count minimization. The question is whether high-level synthesis could take advantage of floorplan information. Given a hierarchical flow based on planning, then block prototyping, which are re-injected at the top-level to refine and adjust the budget, one can envision that high-level synthesis' role will become different:

- Gate synthesis should be very fast so that one can explore the possible tradeoffs during the planning phase.
- It should focus on getting the right logic structure, not on doing gate-level optimization
- It should be incremental, to reflect budget adjustment.
- It may take advantage and/or drive some floorplan information, e.g., pin placement.

### Conclusion

With every process generation, there are more transistors, higher clock frequencies, and additional physical effects to consider. This paper discussed the challenges that need to be addressed during physical implementation in the creation of a production worthy GDSII. A refinement-based flow, which includes clock tree, power routing, and scan chain, with an open cost function, together with physical logic synthesis and optimization, can meet these challenges. Such a flow enables early estimation of congestion and timing, which is

continuously refined up to a physical prototype, where the final implementation results can be accurately predicted. The physical prototype produced by the refinement-based flow restores the design signoff solution lost in the mid-90's.

Physical prototype also enables a top-down budgeting, bottom-up implementation, hierarchical flow. This hierarchical flow allows continuous chip-level validation, budget adjustments, and block changes, throughout the life of the project.

### References

- [1] O. Coudert, A. Narayan, "Subsystem: Logic Optimization", Tech. Report, Monterey Design Systems, Sept. 1998.
- [2] O. Coudert, "Logical and Physical Design: A Flow Perspective", in *Logic Synthesis and Verification*, S. Hassoun & T. Sasao Editors, Kluwer Academic Publishers, Chapter 7, pp. 167--196, 2001.
- [3] G. Karypis, R. Aggarwal, V. Kumar, S. Shekhar, "Multilevel Hypergraph Partitioning: Application in VLSI Domain", in Proc. of 34th DAC, pp. 526-529, June 1997.
- [4] T. Okamoto, J. Cong, "Interconnect Layout Optimization by Simultaneous Steiner Tree Construction and Buffer Insertion", ISPD'96, 1996.
- [5] M. Pedram, N. Bhat, "Layout driven technology mapping", 28th DAC, June 1991.
- [6] R. F. Sproull, I. E. Sutherland, "Logical Effort: Designing for Speed on the Back of an Envelope", in Proc. of IEEE Advanced Research in VLSI Conference", 1991.
- [7] G. Stenz, B.M. Reiss, B.Rohfleisch, F.M. Johannes, "Timing Driven Placement in Interaction with Netlist Transformations", in Proc. of ISPD'97, pp. 36-41, 1997.
- [8] H. Zhou, M. Wong, I-M. Liu, A. Aziz, "Simultaneous Routing and Buffer Insertion with Restrictions on Buffer Locations", 36th DAC, June 1999.