

Chapter 7

LOGICAL AND PHYSICAL DESIGN: A FLOW PERSPECTIVE

Olivier Coudert

Abstract A physical design flow consists of producing a production-worthy layout from a gate-level netlist subject to a set of constraints. This chapter focuses on the problems imposed by shrinking process technologies, and their solutions in the context of design flows with an emphasis on the complex interactions between logic optimization, placement, and routing. The chapter exposes the problems of timing and design closure, signal integrity, design variable dependencies, clock and power/ground routing, and design signoff. It also surveys logical and physical design flows, and describes a refinement-based flow.

7.1. INTRODUCTION

Creating a chip layout worthy of fabrication is achieved through a sequence of steps, coined as a *design flow*. Since the event of logic synthesis in the mid-80's, design flows have been characterized by a clear separation between the logical and the physical domains. Logic synthesis produces a gate-level netlist from an abstract specification (behavioral, RTL). Place and route produces the layout from the gate-level netlist and technology files. Specific tools have been used to create the clock tree and the power/ground (P/G) routing, and to achieve final signoff. Reconsidering the logical aspects during the physical design phase was unnecessary, because timing signoff could be done at the gate level, and signal integrity was rarely an issue.

A flow consisting of logic synthesis followed by place-and-route cannot work with deep submicron (DSM). At $0.18\mu m$ and beyond, interconnect becomes a dominant factor and breaks the dichotomy between the logical and physical domains. First, we can no longer neglect the impact of the interconnect on timing because interconnect dominates gate delays.

Second, increasing design complexity makes managing congestion more challenging. Third, finer geometry, higher clock frequency, and lower voltage produce signal integrity problems. Logical and physical aspects are tied together and can no longer be looked at separately: there is a need for design tools and design flows that enable the simultaneous optimization of the logical and physical aspects of a design.

This chapter discusses logical and physical design from a flow perspective. Section 7.2 explains why the logical and physical variables are interdependent, and why they must be considered together during the physical implementation phase. Section 7.4 first discusses the uncertainty/refinement principle, then presents the details of a refinement-based design flow, a promising approach to solve the problems brought by DSM. The chapter concludes with some thoughts on a full-chip hierarchical design flow.

7.2. LOGICAL AND PHYSICAL DESIGN CHALLENGES

VLSI physical design is the process of producing a GDSII (a mask describing the complete layout that will be fabricated) from a gate-level description while satisfying a set of constraints. The gate-level netlist is a logical description of the circuit in terms of cells (e.g., I/O pads, RAMs, IP blocks, logical gates, flip-flops, etc.) and their interconnections. The constraints include timing (setup, hold, slope, min/max path delays, multicycle paths, false paths), area, place keepouts (regions of the die where cells cannot be placed), and route keepouts (regions of the die through which nets cannot be routed), power consumption, and technology constraints (e.g., maximum output load, electromigration rules). This section presents the challenges that must be addressed by a physical design flow intended for DSM designs.

7.2.1. TIMING CLOSURE AND DESIGN SIGNOFF

Timing closure is the problem of achieving the timing constraints of a design. Until the mid-90's, most of the delay was in the gates, and the net capacitance was a relatively small contributor to the total output load of the gates. Thus the impact of nets on timing could be disregarded. Consequently timing signoff could be done with static timing analysis on a gate-level netlist, right after synthesis (Fig. 7.1, left). From there, the netlist could be handed off to physical implementation.

As process geometries scale down, interconnect becomes the predominant factor in determining the delay. First, the gate delay depends

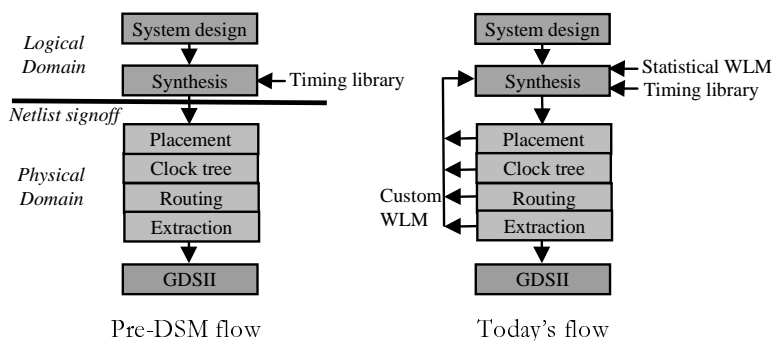


Figure 7.1. Flows until mid-1990's.

mostly on the output capacitance it drives, of which the net capacitance becomes the largest contributor. Second, the delay of long nets, which depends on their capacitance and resistance, becomes larger than gate delays.

Timing signoff is the process of guaranteeing that the timing constraints will be met. Post-synthesis signoff was possible when interconnect contributed less than 20% of the total output capacitance. Now that net capacitance is becoming more dominant with every new process generation, accurate timing estimation without knowing net topologies and placements is not possible. Since timing cannot be realistically estimated from a netlist alone, it is not practical to think in terms of a post-synthesis netlist signoff.

One could attempt to account for interconnect delays by using a statistical wire load model (WLM) based on pin count. The problem is that timing is determined by a *maximum* path delay, and although such a wire load model is actually a good predictor of the *average* wire load, the deviation is so large that timing becomes grossly mispredicted.

Moreover, coupling capacitance becomes more dominant over inter-layer capacitance with every new process technology because the wires are getting much closer to each other and the interconnect aspect ratio of height to width is increasing (Fig. 7.2). In 1999, the coupling capacitance C_c —the lateral coupling capacitance between interconnects on the same layer—was about three times larger than the inter-layer capacitance C_s —the capacitance due to overlapping of interconnect between different layers—. The ratio is projected to increase to five in 2006 (Fig. 7.2). This means that the capacitance of a net cannot be determined without knowing both its route *and* that of its neighbors.

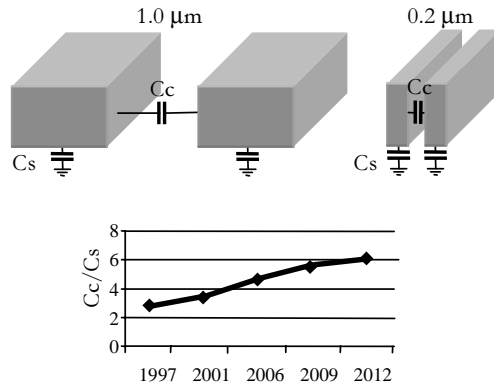


Figure 7.2. Coupling capacitance dominates inter-layer capacitance.

More generally, what is needed is a *design signoff* solution that will validate all the design variables, i.e., not only timing, but also area, congestion, power, and signal integrity.

7.2.2. SIGNAL INTEGRITY

Deep submicron results in a number of physical effects that must be controlled to ensure signal integrity and correct functioning of the chip. 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 induces signal noise. Other physical aspects such as antenna effect, electromigration, self-heating, IR-drop, need to be analyzed and controlled. The next technology generation will need even more detailed analysis, including inductance of chip and packaging, on-chip decoupling, resonance frequency analysis, and soft error correction. Signal integrity problems must be identified as early as possible because it is very costly and time consuming, if not impossible, to fix them during the final implementation stages.

The deep submicron era is causing a shift from timing closure to design closure, where every design variable (timing, area, power, congestion, signal integrity) needs to be monitored and optimized simultaneously when implementing the chip. This era marks the end of the logical and physical dichotomy.

7.2.3. DESIGN VARIABLES INTERDEPENDENCY

Some interdependencies are well known: overall area increases when timing is tighter, power increases as area increases. In the DSM era more complex interdependencies are evolving. The timing/placement interdependency is the most obvious. Without placement, the interconnect information is insufficient to evaluate the timing. Also, every placement change will impact timing. Similarly delay decrease/increase will enable/disable placement changes.

With smaller geometry features, the complexity and the physical resources of the interconnect make congestion (i.e., routing resource requirements) a key factor in the design closure process. Controlling congestion is a challenging problem because of the complex timing/placement/congestion interaction. For example, shortening nets by moving gates and/or straightening routes can help in reducing the delay, but this pulls gates and/or nets closer, creating congestion problems. Conversely, high congestion spots can be relieved by downsizing and/or spreading gates around, or by rerouting nets, which can create new timing problems. Also, long wires that go across the chip cannot be designed without a placement, and the placement itself should know about the global routing of the long wires to account for their contribution to congestion. In addition, more local dependencies are created with area, power, and signal integrity (e.g., crosstalk affects timing). A physical design system with an open cost function is required to actually address all these complex interdependencies.

7.2.4. CLOCK AND P/G ROUTING

In a traditional design flow, clock tree synthesis and power routing are separate tasks. Typically, the clock tree is built once the locations of the sequential elements and gated clocks are known, i.e., after placement. But designing clock trees and P/G networks after placement creates routing problems that occur too late to be fixed. Thus placement must know about clock and P/G routing so that it can account for their contribution to congestion.

Also, the clock tree is becoming denser, spanning a larger chip with a tighter clock period, thus requiring good control of the skew. Fine control of the skew should be used to optimize timing.

The same principle applies to scan chains. The impact in terms of congestion is much smaller, but it is important to re-order the scan chains so that the placement is not overconstrained.

7.2.5. CAPACITY

Designs keep getting bigger and more complex, containing tens of millions of gates. Two capacity problems are emerging. The first is a pure scalability problem: the raw number of objects that need to be managed and processed is stressing memory limits and computational resources (only linear or $n \log n$ algorithms can reasonably be applied to a complete netlist). The second problem is the overall complexity of a chip design, which is often divided into several blocks, with several independent design teams working in parallel, each at a different pace. This problem is inherent to project management, meaning that few very large chips are actually designed flat: the trend is towards more hierarchical designs.

Hierarchical flows pose new problems for physical design, e.g., handling timing constraints between the chip level and the block level, verifying the feasibility of the constraints at the chip level, capturing the chip context when implementing a block, hierarchical chip verification, etc. There are no tools to help the designer to develop a top-down timing budget of the blocks, and no tools to model the interaction (e.g., timing and congestion) between the blocks as they are designed bottom-up. Floorplanning is still a difficult problem, especially when it must be done with timing and congestion considerations.

There are several research efforts to enable complete hierarchical physical flows, but, because of lack of space, we do not address the hierarchy problem and instead focus entirely on flat physical design.

7.3. SURVEY OF CURRENT DESIGN FLOWS

Several solutions have been proposed to solve DSM challenges and to achieve design closure. The first two flows we describe in this section are “stand alone” flows, in the sense that they do not require new technology on top of a classical sequential flow made of synthesis, followed by place-and-route. The last two flows are attempts in more tightly integrating synthesis and place-and-route.

Iterative flow with Custom wire load model. This flow iterates between gate-level synthesis and place-and-route. After place-and-route, if the constraints are not met, the netlist is back-annotated with the actual wire load and re-synthesised (Fig. 7.1, right). Signal integrity problems are usually handled at the detailed routing level.

Place and route data are necessary to determine the timing. Trying to compensate for the lack of these data by driving synthesis with pessimistic assumptions results in over design. Extracting net capacitances

after place and route and feeding them back to synthesis in an attempt to seed synthesis with more realistic capacitance estimations is not practical. This often results in a different netlist, with a distinct placement and routing, thus producing different wire loads and timing. There is no guarantee that this iterative process will eventually converge.

Expecting the backend tools to fix problems that are recognized late in the flow is unrealistic, because the latitude of what can be done at this level is limited. This solution is ineffective for technologies at or below $0.18\mu\text{m}$.

Block-assembly flow. This flow consists of designing small blocks independently, then assembling them. The idea is that a statistical wire load model is sufficiently accurate for small blocks of logic — $50k$ cells has been proposed [32]—. The netlist is divided into blocks such that the intra-block interconnect delay can be neglected or roughly estimated, which enables synthesis to predict the overall delay. Then the blocks are assembled.

There are several problems here. First, dividing a chip into blocks requires time budgeting, and there is no scientific method to come up with an accurate budgeting that can be met both at the block level and at the chip level. This results in a sub-optimal or infeasible netlist. Second, assembly must respect the physical boundaries of the blocks to ensure that the intra-block delays are preserved. This significantly constrains the placement such that timing and/or congestion problems cannot be addressed easily. Third, it is virtually impossible to estimate the inter-block delays, since long interconnects depend on the relative placement of the blocks. Fourth, statistical wire load models may still fail to predict timing accurately, even for small blocks, due to routing congestion. If routes in the block are forced to meander in congested areas, the net capacitances increase substantially. Because congestion impacts wire load model predictability, the overall connectivity of the netlist must be considered, which makes impossible to look at a block alone without understanding the chip-level context it is placed.

Overall, this is essentially a trial and error approach, forcing many iterations. Furthermore, this approach does not even address the problem of signal integrity.

Constant delay synthesis flow. The delay through a logic stage (i.e., a gate and the net it drives) is expressed as a linear function of the gain, which is the ratio of the capacitance driven by the gate to its input pin capacitance. The first step in the constant delay synthesis flow consists of assigning a fixed delay (i.e., a fixed gain) to every logical

stage so that timing constraints are met. The second step consists of placing and routing the netlist while preserving these gains.

Constant delay synthesis is attractive because of its elegance and simplicity, thus enabling fast synthesis. RTL-to-gate synthesis has been proven a good application of constant delay synthesis. However, this elegance is obtained at the cost of ignoring the physical aspects. Delay models must be input-slope dependent, and must distinguish between rising and falling signals. The propagation of a waveform in an interconnect cannot be captured by such a simple model. The gains can be kept constant by adapting the net capacitances and the gate sizes within limited ranges. In practice, this means that the netlist *must* be changed to accommodate the placement/route/timing constraints (e.g., re-buffering), which means that delays must be re-assigned. Constant delay synthesis assumes continuous sizing. Mapping the resulting solution onto a real discrete library can result in a sub-optimal netlist. Also constant delay synthesis relies on convexity assumptions (see Section 7.4.3.1) that are often not true for realistic libraries.

Placement-aware synthesis flow. This kind of flow is a general trend, because synthesis needs placement information to estimate the timing [26, 31]. But the meaning of “aware” is often fuzzy. Gluing synthesis and placement together does not help if routing information is not sufficient, or if the interaction between synthesis and placement is not under control. For example, synthesis may locally increase the area to fix a timing problem, thus creating an overfilled area, to which placement will react by spreading gates around, which will create other timing problems. A strategy leading to convergence is a major problem. Moreover, synthesis and placement working together is clearly not sufficient if one cannot account for congestion and signal integrity, which requires an understanding of routing and physical aspects. In other words, this approach does not go far enough in the integration of the logical and physical domains.

7.4. REFINEMENT-BASED FLOW

As stated in Section 7.2, design closure requires that all design variables need to be considered together (timing, area, power, congestion, clock tree synthesis, P/G routing, scan chain reordering, signal integrity). This section introduces a novel physical design flow based on the concept of refinement. It first discusses the uncertainty/refinement principle. It then describes how to apply this principle during placement, logic optimization, clock tree synthesis, P/G routing, etc. Finally, this section

explains how place, logic optimization, and global route interact, and how to achieve design signoff.

7.4.1. THE UNCERTAINTY/REFINEMENT PRINCIPLE

For any variable x , let \hat{x} be its estimate. Assume that we want to minimize a function $x(p)$ by finding an optimum value of p . Consider the following remarks:

- A parameter x cannot be optimized beyond the range it can be estimated, i.e., an optimization procedure which computes the parameter p producing the best value $\hat{x}(p)$ is worthless if these values are within the uncertainty of \hat{x} .
- The optimization of $x(p)$ should be done with a resolution of p that cannot exceed the uncertainty of \hat{p} , e.g., if we know precisely the radius r of p (i.e., $|p - \hat{p}| \leq r$), then we should look at $x(p)$ by $\pm 2r$ increments of p .

A variable of a design in progress is always estimated with some uncertainty. For example, the timing is fully known only if the routes are fully known. If we have partial placement or/and route information only, then the uncertainty on the net capacitances results in an even larger uncertainty on the timing. Another simple example is IR-drop (voltage drop in the power nets), which can be known only as much as the placement, toggle rates, and P/G routing are known. IR-drop itself leads to an additional uncertainty in timing (Section 7.4.6.2). At the beginning, the only available data are the gate-level netlist, the library, the technology files, the constraints (timing, die area, pre-placed cells and pre-routes, power). Thus unless we have enough information about the placement and routes, it is simply *useless* to do any kind of meaningful timing optimization beyond simple wirelength minimization.

Indeed, if we know the resolution 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. If we augment the resolution of the placement and routing, new design parameters become “visible” and can be optimized in turn.

A flow respecting these observations starts with the initial netlist and physically builds the design, step by step, by increasing the resolution of every parameter simultaneously. Early in this process, there is only an approximate placement and global routing, and the clock tree and P/G network are roughly estimated (remember that we must account for all contributions to the congestion). At the end, there is a fully detailed,

placed and routed netlist, including the completed clock tree and P/G routing. Between these two points, there is a continuous refinement process 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 are the keys to prediction and convergence.

A refinement technique that knows the accuracy of each design measurement can react to a problem at the earliest appropriate time, i.e., as early as possible and only when it becomes meaningful. For instance, as the placement and global route are refined, wirelength and congestion can be estimated and optimized at the very beginning, then timing, then IR-drop, then crosstalk, etc.

In the sequel we discuss various aspects of implementing such a refinement-based flow.

7.4.2. PLACEMENT TECHNIQUES

Placement techniques can be classified roughly in two types: analytical (constructive) placement, and move-based (iterative improvement) placement. Analytical placement uses linear or convex programming to minimize a closed-form, differentiable function. Move-based placement evaluates placement attempts and decides which moves to actually implement. We describe two analytical placement methods, quadratic and force-directed placements; and two constructive methods, simulated annealing and quadrisection-based placement. We then explain why the latter is a good choice for a refinement-based physical design flow.

7.4.2.1 Quadratic placement. Quadratic placement is based on minimizing the sum of the squares of the wirelengths. Every cell has a xy coordinate in the plane, and the cost of a wire between a cell i and j is captured by $(x_i - x_j)^2 + (y_i - y_j)^2$. Thus the problem translates into solving a linear system $Ax + b = 0$, where x is the vector of the movable cell coordinates, A is the connectivity matrix of the cells, and b is the vector of the fixed cell coordinates (e.g., I/O pads).

Without fixed cells, the solution is trivial, with all the cells on top of each other. Even with fixed pads, the solution usually shows most of the cells overlapping. Thus quadratic placement iterates squared wirelength minimization and bi-partitioning [21]. Bi-partitioning divides the set of

cells in two and assigns each part to one area of the chip. Then squared wirelength minimization is applied to each partition, and the process is iterated until a legal placement can be inferred.

Quadratic placement is very fast and can handle large designs. However, it aims at minimizing the squares of the wirelengths, not the actual wirelengths. It is also not suitable for timing-driven placement. Net weighting can be used to emphasize critical nets, but the criticality of a net may change as the placement changes, and estimating the criticality of a net at the top level, with very fuzzy placement information, is unrealistic. Also there is no good solution to handle congestion during quadratic placement. However, because of its speed, quadratic placement is often used to seed a more sophisticated placement method.

7.4.2.2 Force-directed placement. Force-directed placement is another analytical method. The principle is to include, in the equations capturing the wirelength, a term that penalizes for overlapping cells, so that a balance can be established between minimizing the wirelength and yielding a legal placement [10]. Various formulations of this principle can be used, and it is usually solved by conjugate gradient iterations or other convex programming techniques.

Force-directed placement is slower than quadratic placement, but gives significantly better results. A key aspect is that some force-directed placement algorithms can place objects with a large range of sizes, enabling mixed block, megacell, and standard cell placement. However, there is a limit to what can be expressed with an analytical function. Some costs are thus hard to capture, and the tuning of the “repulsive” terms can be difficult.

7.4.2.3 Simulated annealing. Simulated annealing [30] is based on move-and-cost. A move consists of moving an object from one location to another (it can also be a swap, where two object locations are exchanged). A move is evaluated and accepted if it improves the cost. A move that degrades the cost is accepted with a probability usually expressed as $e^{-\frac{|\Delta cost|}{T}}$, where $\Delta cost$ is the fitness of the move (the delta cost resulting from moving it), and T is a global parameter called temperature. At the beginning, the temperature is large, thus the probability of accepting a move degrading the cost is high. As placement progresses, the temperature is lowered, and fewer “bad” moves are accepted. Accepting cost-degrading moves allows exploring larger solution spaces and avoiding local minima.

Simulated annealing has a completely open cost function, and it can produce globally optimum solutions. However, it is an extremely slow

process, limiting its application to small sets of objects. Due to its optimality potential and speed limit, it is only used for end case placement, e.g., detailed placement.

7.4.2.4 Quadrisection and cluster-move based placement.

Bisection placement iterates min-cut balanced partitioning and move-based placement. The netlist is partitioned in two sets of similar area, with as few nets spanning the two sets as possible [4]. Then cells are moved or swapped between partitions to reduce the cost. When a local minimum is reached, each set is partitioned, and moves are applied between the resulting partitions. Quadrisection is the same process when the partitioning is made of four partitions instead of two. Experience shows that it is better suited for 2D placement.

Moving one cell at a time is far too slow to process a real design. A dramatic speedup is obtained by allowing groups of cells to be moved together. This method is also known as multilevel hypergraph partitioning [20]. A hierarchical tree of clusters of cells is first built from the netlist. The top of the tree is made of large clusters of cells, and the leaves of the tree are the individual cells. The cluster tree is built such that connectivity is reduced, area is balanced, and functional/physical hierarchical constraints, if any, are satisfied. It can be derived from a mix of balanced multi-way min-cut partitioning (top-down) and topology driven netlist clustering (bottom-up). Then the clusters are placed and moved in a four-way min-cut quadrisection (Fig. 7.3). When a local minimum is reached, the netlist is re-clustered, each partition is quadrisected, and the whole process is iterated (Fig. 7.3).

This placement technique has a fully open cost function, and it produces good quality solutions. Also, since it is move based, it is more suitable to timing-driven placement, provided that the partition resolution is sufficient to estimate the timing. This move-based method with an open cost function allows simultaneous optimization of all design aspects (including timing, congestion, wirelength, and crosstalk).

7.4.2.5 Placement refinement for physical design.

A physical design flow must be aware of all the design variables. This requires an open cost function, which can capture timing, area, power, congestion, etc. This immediately rules out quadratic placement, which is only good at reducing squared wire length and cannot have any understanding of critical paths or high congestion areas. Force-directed placement methods rely on analytical expressions that can be minimized using convex programming or by iterating conjugate gradients. They are slower and more difficult to tune than quadratic placement but capture

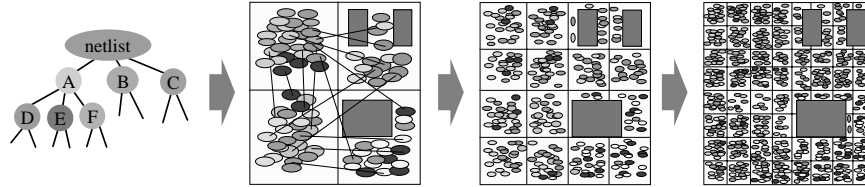


Figure 7.3. Placement refinement.

far more elaborate costs than just squared wire length. However, their cost function is semi-open because a highly non-monotonic, non-smooth cost function cannot be easily included. Simulated annealing has an open cost function and can produce excellent solutions, but is extremely slow and just cannot be used on a real netlist.

A placement technique based on moving clusters across partitions allows an efficient two-dimensional placement with the benefit of an open cost function. Moving a cluster of cells, instead of one individual cell at a time, makes the placer fast. Using a fully open cost function makes the placer aware of the impact that moving a cluster of cells will have on timing and congestion. Also, the flexibility of the cost function allows the placer to account for the impact of the clock tree, P/G routing, and scan chains on the quality of the design (e.g., timing, congestion, IR-drop).

Another advantage of a placement based on refining the placement of objects in smaller and smaller bins, is that it naturally allows several views of the chip. At the top level, there is a very rough placement in a few bins. At the bottom level, there is an accurate placement with a few dozen cells per bin, but it is still flexible (i.e., no precise xy coordinate yet). In between there are placement views that can be used to accommodate more or less disruptive netlist changes. One of the most immediate applications is to accommodate netlist changes performed by logic synthesis and optimization [16, 5]. Another crucial application in the context of complex chip design is to accommodate ECO (Engineering Change Order) changes. Most ECO's are still manual and consequently very local (e.g., changing/removing/adding a gate, connecting/disconnecting a pin to/from a neighbor net). However, as the design complexity increases, so does the level of abstraction used by the designer. An ECO change at the RTL level can affect hundreds of cells, and it is even more disruptive at the behavioral level. Having several levels of placement resolutions eases dramatically the integration of disruptive ECO.

In the rest of this discussion, we will assume the use of a move-based quadrisection cluster placer. The placement is refined into smaller and smaller bins. As the number of quadrisections increases, the place and route information is more detailed, and more sophisticated estimation models using more detailed resolution can be used. Placement is first driven by wire length and congestion until it reaches a resolution for which there is enough information for meaningful timing estimation and optimization. At this point, and from this point *only*, logic optimization to fix timing and congestion problems can be done. Placement and logic optimization then proceed together, first addressing timing and congestion, then considering crosstalk and signal integrity as the refinement allows.

7.4.3. LOGIC OPTIMIZATION

The gate-level netlist is initially synthesized with no placement information, and therefore with a crude estimation of delays. Timing becomes meaningful when enough placement information is available. Only at this moment the logic can be revisited with a good understanding of timing, as well as congestion and power.

Physical logic synthesis and optimization consists of changing 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 has to properly interact with the placer and router.

Physical logic synthesis and optimization has the following requirements. It needs enough place and route information. Placement and routing must be able to accommodate for the logic changes. It is by nature local, because it goes together with placement refinement, and thus it should not disrupt the logic to a scale larger than the placement resolution. It should take advantage of accurate physical data, in particular for timing (input slope dependence, output load dependence, interconnect delay, rising/falling signal delay dependence, crosstalk awareness). It also needs an efficient incremental static timing analyzer, since many local logic transformations may be needed and must be reflected at the global level. The static timing analyzer must support multi-cycle paths, false paths, transparent latches, and be crosstalk aware (see Section 7.4.6.1).

The delay model accuracy must match the resolution of the placement. The simplest delay metric is in terms of total net capacitance. In this model, all fanouts of a net have the same delay, since the net is

approximated as an equipotential surface. It is valid as long as the metal resistance is negligible with respect to the gate resistance. With shrinking features, the metal resistance per unit of length increases, while the gate output resistance decreases. Metal resistance cannot be ignored for an increasing number of nets. A net must be treated as a distributed RC network with different delays at its fanouts. Elmore delay can be used as a first approximation, but more detailed models are required when metal resistance increases.

Logic optimization can significantly move cells and locally modify the timing and area distribution. As said above, logic optimization should not disrupt the netlist to a scale larger than the placement resolution (i.e., the size of a bin) to ensure that placement can accommodate the disruption. Thus as placement resolution increases, the logic optimization becomes less disruptive. At higher levels of quadrisection, the placement is very flexible, and aggressive re-synthesis and technology mapping can be used. At lower levels, only local optimizations, e.g., sizing and buffering, or very focused re-synthesis, should be authorized. After detailed placement, only restricted sizing can be applied.

Among the physical logic synthesis and optimization techniques, one can distinguish:

- Load and driver strength optimization. This includes gate sizing, buffering, pins swapping, gate cloning.
- Timing boundary shifting. This includes transparent latches (i.e., cycle stealing), useful skew, and retiming.
- Resynthesis and technology remapping.
- Redundancy-based optimization
- Area and/or power recovery.

These transformations have to decide the placement of the cells. Some new logic transformations specific to physical design, such as synthesis for congestion relief [8], or logic optimization for signal integrity, are emerging. In the sequel we discuss some logic optimization methods in the context of physical design: sizing, buffering, resynthesis and technology mapping.

7.4.3.1 Sizing. The goal of gate sizing is to determine optimum sizes for the gates so that the circuit meets the delay constraints (slope, setup, and hold) with the least area/power cost. A larger gate will have a higher drive strength (lower resistance) and hence it can charge/discharge output capacitances faster. However, it usually also

has a higher input capacitance. This results in the preceding gate seeing a larger capacitive load and thus suffering an increased delay. Downsizing a gate off the critical path, but driven by a cell on the critical path, may result in decreasing the capacitance seen by the driver, but it can also slow down the off-critical path to the point where it becomes critical. Sizing thus requires a careful balancing of these conflicting effects. An optimal solution will require the coordination of the correct sizes of all the gates along *and* off-critical paths.

We can consider a global solution [6], where the sizes for all the gates in some critical section are determined simultaneously. Analytical techniques has been proposed for sizing in the continuous domain (e.g., linear programming [3], posynomial [11], convex programming [17, 29]), and various attempts have been made to use these results with discrete sizes in a cell library based design style [15, 2]. These method can be practical at the gate level, where full accuracy is not an issue, or at the transistor level, when the models are well characterized and have nice convex properties.

The theory of constant effort [27] and its more usable approximation, constant delay [13], provide a way to select cell sizes directly for each stage. For example, the optimal delay on a fanout-free path is obtained by distributing the effort evenly among the logical stages. This means that if the delay of the whole path is fixed, then the delay of every stage must be kept constant. Thus, cell sizes can be selected to match this constraint by visiting all the gates in a topological order, starting from the outputs. See Section 7.3 for some limitations of constant delay sizing.

Sizing at the physical level must use the most accurate delay model, including interconnect delay, and must consider the input slope effect, as well as the difference between falling and rising delays. This cannot always be captured by analytical techniques. Also these techniques assume that the input pin capacitances of a gate can be expressed as a smooth function (usually linear) of the gate's size (i.e., drive strength). Unfortunately this assumption is often broken: input pin capacitance does not always nicely correlate with gate size, and sometimes it is not even a convex function. It is possible to design libraries that specifically meet these requirements, but library design and validation is a huge burden and investment.

Alternatively, discrete sizing can be done using global search techniques [7]. These techniques attempt to reach a global optimum through a sequence of local moves, i.e., single-gate changes. While these are computationally expensive, they can take advantage of very accurate delay models, they are not limited by assumptions needed by analytical tech-

niques (e.g., a well-built library with convex input pin capacitances), they can enforce complex constraints by rejecting moves that violate them (e.g., validity of the placement), and they can simultaneously combine sizing with other local logic optimizations (e.g., gate placement, buffering, pin swapping). They have been shown to provide good results for discrete libraries. They are particularly effective in the context of physical design, since they can focus on small critical sections, use the most accurate delay models—including interconnect delay—and find a small sequence of moves that meets timing constraints while maintaining the validity of the placement.

Analytical sizing methods based on simple convex delay models are global and fast. Discrete sizing methods are slower, but very accurate and focused. Both methods should be used by physical synthesis, driven by the size of the bin during the placement refinement, because the bin size determines the extent of the authorized netlist change, and the accuracy of the physical information.

7.4.3.2 Buffering. Buffering serves multiple functions:

- Long wires result in signal attenuation (slope degradation). Buffers (a.k.a. repeaters in this context) are used to restore signal levels.
- A chain of one or more buffers can be used to increase the drive strength for a gate that is driving a large load.
- Buffers can be used to shield a critical path from a high-load off-critical path. The buffer is used to drive the off-critical path load so that the driver on the critical path sees only the buffer's input pin capacitance in place of the high load.

It is possible to come up with ad-hoc solutions for each of the above cases. For example, repeaters can be added at fixed wirelength intervals determined by the technology. However, critical nets often need to be buffered to satisfy more than one of the above listed requirements. Thus, we prefer algorithmic solutions that balance the attenuation, drive strength, and shielding requirements. This problem is hopelessly NP-hard. An interesting solution exists for the case when the topology of the buffer tree is fixed and the potential buffer sites are fixed. This is the case when the global route for the net is already determined and the buffer tree must follow this route. A dynamic programming algorithm, using an Elmore delay model for the interconnect, finds an optimal solution in polynomial time [12]. Various attempts have been made to overcome the two limitations of this algorithm (the fact that the topology is already fixed, and that the Elmore delay model does not accurately cap-

ture the resistive effects of DSM technologies). The former is considered to be more of a problem, because fixing the topology can severely limit the shielding possibilities and lead to overall sub-optimal solutions.

What is needed is the ability to determine the net route as part of the buffering solution. Some attempts have been made to develop heuristic solutions [23, 28], including adding additional degrees of freedom like wire and driver sizing. These techniques give better solutions than the fixed topology approach. Various constraints must be handled when routing and buffering the net, like routing blockages and restricted buffer locations due to place keepouts [33, 18].

Determining the optimal buffer tree for a given net is only one part of the complete buffering problem. Given that buffering a given net can change the constraints (required time/slack, load) on the pins of another net, the final solution is sensitive to the order in which the nets are visited. In addition, once a net is buffered, the gates may no longer be optimally sized. Resizing gates before the next net is buffered can modify the buffering problem. Researchers have considered combining sizing and buffering into a single step [19], but again this problem is very complex and far from being solved.

7.4.3.3 Resynthesis and technology remapping. Technology mapping attempts to find the best selection of cells from a given cell library to meet a given delay constraint with the least area/power. Technology remapping during physical synthesis attempts to find a better mapping by using existing physical information for determining interconnect capacitance and delay. Technology mapping has been well studied [14]. Knowing where to apply it is the key. The challenge is to work on sections small enough to not significantly disturb the placement, yet significant enough to improve the design. Another challenge is to determine where to place the new cells created during the remapping phase [22]. Some simple solutions based on fixed boundary locations can be used during the mapping itself, with a clean-up step to make the placement legal [24].

Logic restructuring is the strongest in the suite of logic optimization techniques because it can significantly change the structure of the netlist. This makes it the most difficult to apply in a physical design flow where the changes are expected to be small to maintain the validity of the placement. However, the basic ideas in restructuring can still be used to improve the timing and congestion properties of the netlist as long as the changes are focused on key sections and the modifications are within the space of acceptable changes, e.g., the changes do not violate capacity/congestion constraints for various physical regions of the

design. The restructuring techniques themselves range from gate collapsing/decomposition (based on commutative, associative, and distributive properties of logic functions [14]), to sophisticated Boolean resynthesis (e.g., BDD-based). Again the challenge is knowing where to apply them, and maintaining the constraints imposed by the existing physical design. Several researchers have worked on this [25], however, with no dominant technique emerging.

Timing driven logic resynthesis based on arrival times of critical signals can be used—the basic idea is using late signals as close to the outputs as possible—. However, one also wants to keep together signals that are physically close to each other. This means that in addition to its arrival time, the location of a signal must be taken into account as the Boolean function is synthesized.

Resynthesis technique can help relieving congestion. An innovative technique consists of designing the interconnect *before* the logic. The logic is then synthesized by following the Boolean constraints established by the interconnect [8]. This problem does not always have a solution; thus either the interconnect must have enough flexibility to enable the synthesis of the logic, or the logic itself must be restricted up front to guarantee a feasible solution.

7.4.4. CLOCK TREE SYNTHESIS

Clock tree synthesis must be part of the refinement process, because it significantly affects congestion, power dissipation, and timing via clock skew. Targeting zero-skew clock trees has no justification but historical. Instead, skew can be used to optimize the timing. Furthermore, a non zero-skew clock tree ensures that all the sequential elements will not toggle at the same time, thus reducing undesirable peak power.

After some level of quadrisection, the distribution of the sequential elements and gated clocks in the bins will not change substantially. At this level one can determine the amount of routing and buffering needed to carry the clock signal, and the trunk of the clock tree can be built (Fig. 7.4, left). From this point these resources are accounted for by placement for congestion. As the partition size decreases and the detail increases, the clock tree is refined and its resources are updated. This leads to no congestion surprises at the end, and it gives tight control over clock skew requirements, since the placement is flexible enough so that clock pins with common skew can be grouped together. It also allows a fine control of the skew for timing optimization, because the critical paths are continuously monitored.

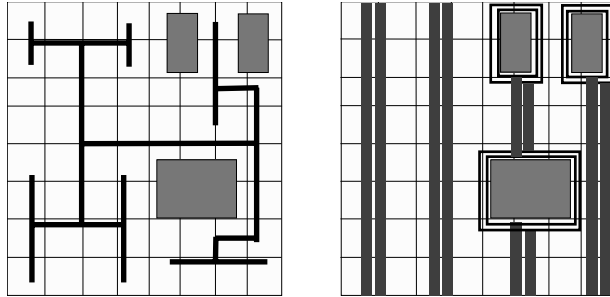


Figure 7.4. Clock tree and P/G network refinement.

Note that the scan chains can be similarly refined, un-stitched and re-stitched to accommodate the placement of sequential elements and to minimize the congestion while still meeting the scan ordering constraints.

7.4.5. POWER/GROUND ROUTING

The power/ground network can have a huge impact on congestion. Initially power routing is performed according to a user-defined routing topology (e.g., chip ring, power grid). At some level of the quadrisection, IR-drop analysis can be done to check the reliability of the P/G network (Fig. 7.4, right). This assesses the quality and integrity of the power routing, because the power rail currents will not change much as the placement is refined. The power consumption depends on net capacitances and toggle rates (number of transitions on a net per unit of time). The switching activity can be obtained by an external simulation (e.g., VCD file), or by an on-the-fly simulation (slow, but accurate), or by probabilistic analysis (fast, but easily misleading). With the distribution of the current sources in the bins, one can extract a power network that is simulated to produce a IR-drop map. This helps in adjusting the power grid, since at this level the placement is flexible enough to accommodate for adding and/or widening power stripes.

7.4.6. SIGNAL INTEGRITY

Signal integrity refers to the many effects that cause the design to malfunction due to the distortion of the signal waveform. These effects were not important in the previous generation of designs. They have become key issues for the correct functioning and performance of DSM chips. In this section we identify the following signal integrity effects

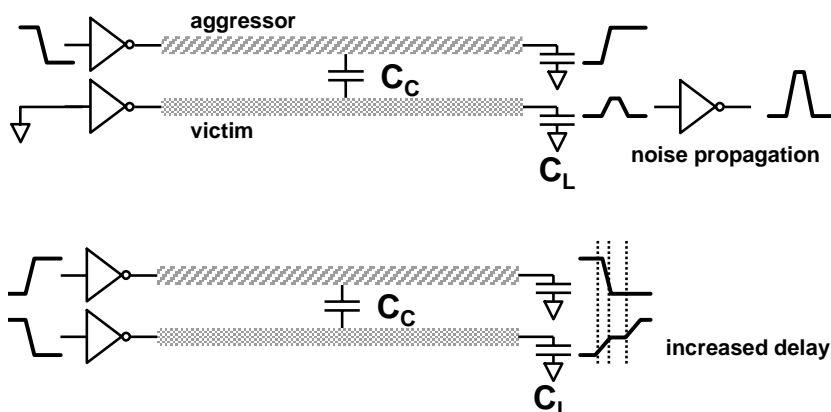


Figure 7.5. Noise and delay crosstalk effects.

and outline possible solutions to address them: crosstalk for delay and noise; IR drop; electromigration; and inductance.

7.4.6.1 Crosstalk. When a signal switches, the voltage waveform of a neighboring net may be affected due to cross-coupling capacitance between interconnects. This effect is called crosstalk. Crosstalk can hurt both timing and functionality. It was not an issue with designs of $0.5\mu\text{m}$ or larger line-widths, but as the technology decreases to $0.25\mu\text{m}$ and below, the coupling capacitance becomes the dominant factor (Section 7.2.1), and induces crosstalk effects that cannot be ignored.

Crosstalk analysis determines the noise induced on a net (the victim) by its switching neighboring nets (the aggressors). Analysis can be a static or dynamic. When the victim net is quiescent, or if there is a separation of the switching windows of the victim net from the aggressor net, crosstalk induces a noise that can be analyzed statically. If the aggressor net causes enough voltage variation in the victim net to affect its digital state (i.e., from logical 1 to 0, or vice-versa), the noise propagates and can eventually be latched by a flip-flop to produce a functional fault (Fig. 7.5, top). When the switching windows of the aggressor and victim nets overlap, crosstalk delays (respectively speeds up) the victim net if the aggressor net switches in the opposite (respectively same) direction, which may cause setup (respectively hold) problems (Fig. 7.5, bottom).

It has been proposed to estimate the crosstalk effect by replacing the cross-coupling capacitance C_c with a grounded capacitance of $2 \times C_c$ for worst case (respectively $0 \times C_c$ for best case). This method is

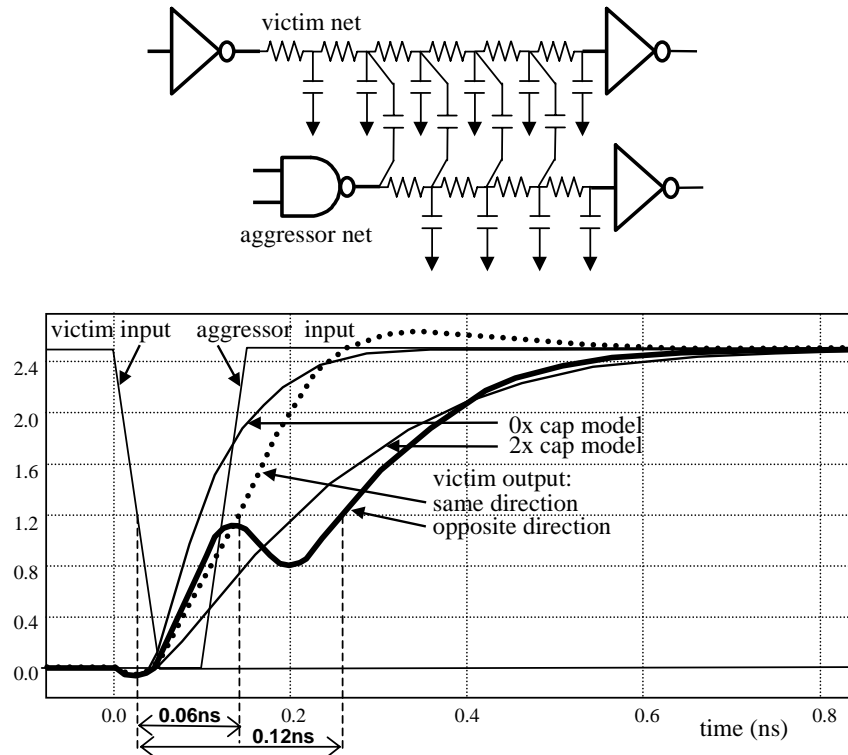


Figure 7.6. RC extraction for crosstalk analysis.

oversimplistic. First, taking $2 \times C_c$ for worst case is *not* an upper bound on the delay induced by opposite direction switching. Second, it does not capture the *dynamic* effect of crosstalk. Such a method simply cannot be used for an accurate timing signoff. Fig. 7.6, bottom, shows the actual waveforms of the victim net for an aggressor switching $0.1ns$ after the victim signal, both for the same and opposite directions. It clearly shows that the waveforms obtained by considering a static $2 \times C_c$ and $0 \times C_c$ grounded capacitance are not good approximations. The actual waveforms show that the victim net's delay is affected on a range of $0.06ns$, which is substantial. If the aggressor net switching window is closer to the victim net's, this range is larger, and it decreases as the window are further away from each other.

A solution consists of a static timing analyzer that generates the switching-time windows for each net [1], together with an accurate cross-coupling RC extractor (Fig. 7.6, top). The dynamic effect of crosstalk

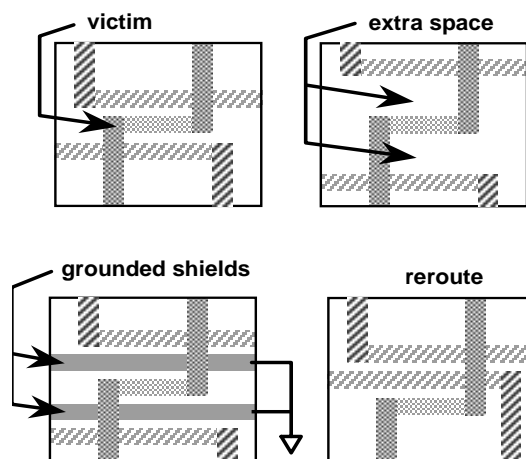


Figure 7.7. Fixing cross-talk at DR.

can thus be accurately analyzed. The same coupling-effect analysis is used for noise analysis. Instead of evaluating the crosstalk contribution to path delay, the waveform calculator determines the amplitude of the induced noise on the victim net. If the noise is larger than some threshold associated with the cell driven by the net, a violation is generated.

Typically, crosstalk analysis is done at the post layout stage (e.g., using a transistor level simulator like SPICE). Then the router tries to fix the problems, e.g., by spacing and/or shielding the afflicted nets, or by switching them to different layers (Fig. 7.7). This can require a major rip-up and re-route effort that has no guarantee to be automatically feasible (the reality is that manual fixes are required). Iterating such a tedious post-layout analysis and fixing process can result in a long design time.

Post-layout fixing is a costly process and may not converge, so crosstalk needs to be addressed as early as possible in the flow. The router can be constrained to avoid crosstalk effects in the first place, e.g., forbidding a maximum length of parallel routing between any pair of nets. However, this method is based on empirical data and does not reflect the physics of crosstalk, which must include signal-switching window dependency. The router ends up over-constrained, leading to unresolvable congestion problems.

There are attempts at implementing crosstalk avoidance during placement and global routing. The idea is that even if detailed routing is not available, the signal switching windows can be used statistically to iden-

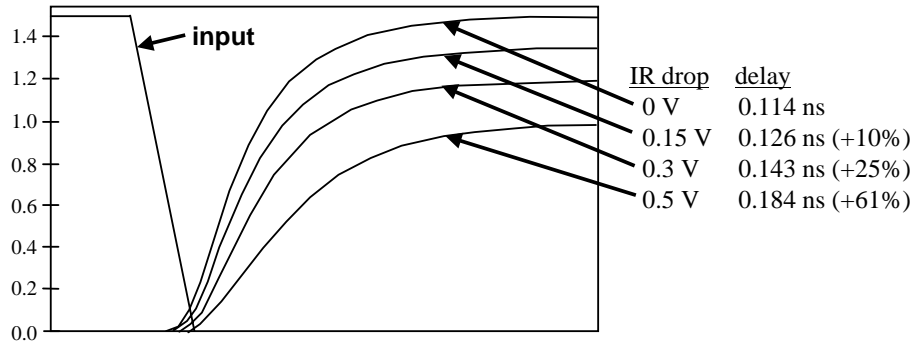


Figure 7.8. IR drop causes delay.

tify the potential problem nets. The problem nets are given more white space during global routing so that the detailed routing has enough resources to perform spacing, shielding, or re-routing and automatically fix crosstalk problems. Also this requires the detailed router to have gridless capabilities with variable width and variable spacing so that crosstalk issues can be addressed effectively.

7.4.6.2 IR-Drop. IR Drop is the problem of voltage change on the power and ground supply lines due to high current flowing through the P/G resistive network. When the voltage drop (respectively rise) in the power (respectively ground) network becomes excessive, this causes delays in gates that can produce timing violations and functionality problems. It also causes unreliable operation because of smaller noise margins. As an example, 1A of current flowing through 1Ω of interconnect causes a 1V drop, which is 2/3 of the 1.5V power supply value. Fig. 7.8 shows the effect of voltage drop at VDD on the performance of a buffer.

IR drop becomes more critical for DSM designs because (1) there are more devices, thus a higher current demand; (2) the wire and contact/via resistance increases because of narrower wires and fewer contacts/vias; and (3) the supply voltage is decreased to 1.5 volts and below.

For many designs, IR drop is being addressed by over-designing the power network with wide power buses and multiple power meshes. However, this severely reduces the available routing resource and is likely to cause routing congestion problems.

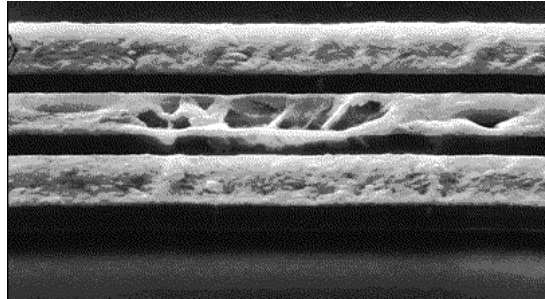


Figure 7.9. Net failure due to electromigration.

An accurate IR drop analysis is done with a transistor level simulation that computes the dynamic current flows. This is a costly process, and at too low a level to allow fixing problems. The simulation can be done at the cell level using cell-level power models, an RC model for the interconnect, and their switching directions and frequencies. This produces an average current per cell, from which the average voltage drop can be approximated with the resistance model of the power and ground network. Although less accurate, this method can identify regions with high voltage drops.

When the level of quadrisection is fine enough, the interconnect loading can be obtained accurately while the placement has enough flexibility to accommodate the power routing change. Optimizing the power routing for both IR-drop and congestion can be done at this level. The currents of the cells in a bin are accumulated and represented as a single current source to the power grid. A fast simulation is then used to evaluate the voltage drop so that the power network can be adjusted accordingly. Power stripes can be narrowed or suppressed to free some routing resources, or widened and augmented to meet the IR drop specification.

7.4.6.3 Electromigration. Increased current density through interconnect for an extended period of time causes resistance to increase. This results in self-heating and metal disintegration, which creates an open or short in the circuit (Fig. 7.9). This effect is called electromigration (EM), and was not a problem until the advent of DSM. As cir-

cuits get faster and bigger, more currents flow through the interconnects, which at the same time are getting narrower with every new generation. The current density (the amount of current flowing per unit area) increases super-linearly with every new DSM generation. It is no longer feasible for manufacturing to provide enough cross-section area on the interconnects to guarantee “net integrity” at all line widths.

EM has traditionally been addressed by over-designing with wide power buses, which is where most of the EM issues are expected. However, over-designing power routing may cause congestion problems and is no longer acceptable. Also, electromigration effects have become important for clock trees, and will affect signal nets in the future. For clock and signal nets, there may not be enough contacts/vias to sustain the current through the interconnect. A solution which will provide sufficient interconnect widths without excessive over-design is necessary.

Tools that compute the current densities from the layout are used to analyze EM problems, which are then fixed manually by widening the problem nets. However, this requires considerable expertise, and the extra space necessary for widening the nets may not be available.

It is possible to address EM much earlier in the design flow, by calculating the required width of the interconnect as placement and routing are refined. Once the placement is accurate enough for a good estimation of the net capacitances, one can calculate the current flow in these nets, which together with the switching activity of the nets, enables electromigration analysis. The interconnect width and via count needed to support the current is identified at each level of quadrisection. These routing resources are then allocated by the global router. Consequently, the detailed router will be able to satisfy the electromigration routing requirements together with other requirements.

7.4.6.4 Inductance. As clock frequency of the design increases to above 500 MHz with process technologies of $0.13\mu m$ and below, inductance becomes an important factor for long nets with high frequency signals. On-chip self inductance affects long nets, and mutual inductance affects nets running in parallel for long distances. Clock nets should be analyzed for self inductance, and bus signals for mutual inductance.

A common approach to address inductance consists of sandwiching the signal layers between power planes (Fig. 7.10, top left). However, this is no longer used due to high manufacturing costs and high power consumption. An easier way to limit the inductance effects is to provide shorter current return paths for the high frequency signals. For clock networks, the current return path should be a ground clock shield running parallel to the clock net in the same layer (Fig. 7.10, top right). For

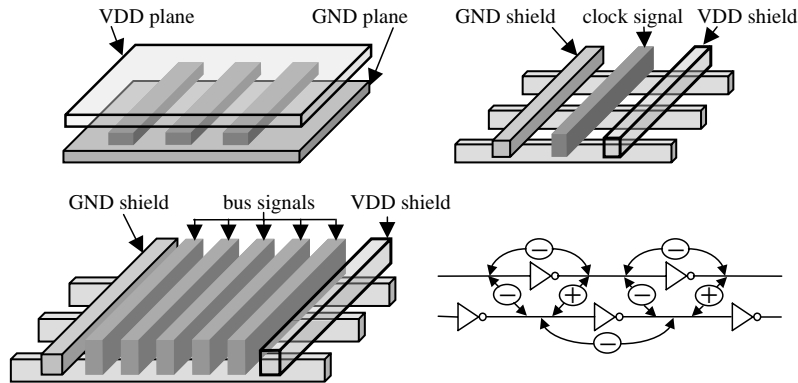


Figure 7.10. Inductance avoidance.

buses, the common practice is to insert a ground wire every 4 to 8 bus signals (Fig. 7.10, bottom left). For pairs of long wires running in parallel, staggered inverters cancel the mutual inductive effects (Fig. 7.10, bottom right).

On-chip inductance is a difficult problem without a clear emerging solution, and it is an area of active research. Even though inductance today affects only the clock tree, signal nets will be affected in the future as the clock frequency rises. New global and detailed routing strategies will be needed to handle this effect.

7.4.7. ROUTING

One of the most important requirements to achieve good routing is the correlation between the global and detailed routers. The detailed router must finalize what the global router predicted, and the routing resources allocated by the global router must actually be available to the detailed router. It is unrealistic to use two uncoupled global and detailed routers, since congestion is a dominant factor in DSM design.

The global router must be timing, congestion and crosstalk driven, and it must support multiple widths and spacing. Since at the beginning it is impossible to know the route of long wires, one can use *probabilistic* route to *smear* the net in the region where it is likely to lie. As the placement is refined and timing models become more accurate, this probabilistic region shrinks, and eventually the net can be fully defined.

The detailed router should support both gridded (for speed) and gridless (for detailed optimization) modes. It must support variable wire width, e.g., for delay optimization. It must enforce numerous DSM

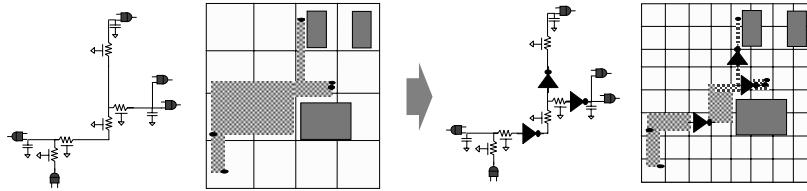


Figure 7.11. Placement/synthesis/routing interaction.

routing rules, e.g., metal/via antenna, end-of-line, minimum area, via array generation for fat-wire connections, and variable spacing based on width. Also it must have timing, congestion, crosstalk, electromigration awareness at all times (e.g., during layer assignment and wire sizing).

7.4.8. PLACE/SYNTHESIS/ROUTE INTERACTION

Fig. 7.11 illustrates one aspect of the placement, routing, congestion, timing, and synthesis interaction. At the beginning, a probabilistic route is generated that spans the area the net will likely lie in. As placement is refined, it takes into account the congestion produced by the smeared route. When synthesis decides to modify the netlist, e.g., by adding buffers to fix a timing problem, the contribution of these buffers to the congestion is also taken into account. The global route is consequently refined, seeded here by the placement and the buffers introduced by logic optimization.

7.4.9. DESIGN SIGNOFF

As the level of quadrisections increases, the layout is known with more detail, and the design variables (area, congestion, timing, power, signal integrity) are estimated more accurately until their optimization becomes meaningful. From this point, they start to be optimized along with the other variables. Eventually one reaches a point where the layout is known with sufficient detail to be able to accurately predict the outcome of the implementation process (final GDSII). The state of this design is a *physical prototype*. Although it occurs well before the design is fully placed and routed, a physical prototype corresponds to the moment where one can truly deliver an actual design of signoff quality to the backend tool.

7.5. CONCLUSION & PERSPECTIVE

Every aspect of the design complexity keeps increasing: more transistors, higher clock frequencies, and more pronounced physical effects. We discussed the challenges that need to be addressed during the physical implementation from gate-level netlists to a production worthy GDSII. We explained how a flow based on placement and route refinement (including clock tree, power routing, and scan chains) with an open cost function, together with physical logic synthesis and optimization, meets some of these challenges. It enables early estimation and optimization of congestion, timing, and physical effects when the design has enough flexibility to accommodate perturbations produced by the optimization procedures: the physical prototype restores the timing signoff lost in the mid-1990's.

The interconnect centric DSM era raises new problems, where placement, routing, and logic optimization are tightly interdependent. Hierarchical design is a necessary direction to handle the capacity and project management problems of multi-million gate designs. This creates new problems, like chip- and block-level physical contexts, and timing abstraction. In essence, the design process above gate-level needs to be more placement aware. At the same time, RTL-to-gate synthesis should not over-optimize a netlist with no place and route data. This leads to a very different full-chip design flow that consists of the following steps:

- *Behavioral synthesis together with floorplanning.* Several scenarios can be explored, with a tradeoff between the block sizes and shapes, the corresponding floorplan, and the resulting chip performances. As a part of floorplanning, the chip is broken up into hierarchical blocks, which include physical and timing constraints sufficient to allow independent physical implementation. Also top-level routing and pin assignment on the block is done to exchange signals between blocks.
- *Fast RTL-to-gate synthesis of the blocks.* The focus is on finding the best logic *structure*, not on having a fully sized and buffered netlist. It could be a grossly mapped netlist, since at this point it does not make much sense to push timing optimization beyond a simple model, e.g., constant delay.
- *Logical and physical design of the blocks.* This is the process described in this chapter. After some placement refinement, enough interconnect information is known to allow resynthesis, remapping, resizing, rebuffering, etc. After signoff on the physical prototype,

the physical implementation of a block within its abstracted chip context is then finalized using the same refinement process.

- *Chip assembly.* Block abstractions and glue logic are assembled using the same refinement process. Note that several levels of block abstractions can be used (from black box to GDSII), which enables early chip-level validation while the blocks are being designed. Final extraction and verification would follow.

To realize such a flow, some of the following questions must be answered:

- What should drive a hierarchical design flow: floorplaning or synthesis?
- How do we perform timing-driven floorplaning?
- How do we re-define gate-level synthesis to facilitate physical optimization?
- How do we derive block-level timing budgets?
- How do we accurately describe the chip context during block implementation (parasitics, timing exceptions, clock tree constraints)?
- Must we define new clock tree design methodologies in such a flow?
- What is the verification and test methodology, functional and physical, in such a flow?

References

- [1] R. Arunachalam, K. Rajagopal, L. Pileggi, “TACO: Timing Analysis with Coupling”, Proc. of *DAC'2000*, June 2000.
- [2] F. Beeftink, P. Kudva, D. Kung, L. Stok, “Gate-Size Selection for Standard Cell Libraries”, Proc. of *ICCAD'98*, pp. 545–550, Nov. 1998.
- [3] M. Berkelaar, J. Jess, “Gate Sizing in MOS Digital Circuits with Linear Programming”, Proc. of *EDAC'90*, pp. 217–221, 1990.
- [4] M. A. Breuer, “A Class of Min-cut Placement Algorithms”, Proc. of *14th DAC*, pp. 284–290, 1977.
- [5] R. Carragher, R. Murgai, S. Chakraborty, M. Prasad, A. Srivastava, N. Vemuri, “Layout-driven Logic Optimization”, in Proc. of *IWLS'2000*, pp. 270–276, May 2000.
- [6] O. Coudert, R. Haddad, S. Manne, “New Algorithms for Gate Sizing: A Comparative Study”, Proc. of *33rd DAC*, pp. 734–739, June 1996.

- [7] O. Coudert, "Gate Sizing for Constrained Delay/Power/Area Optimization", in *IEEE Trans. on VLSI Systems, Special Issue on Low Power Electronics and Design*, 5-4, pp. 465–472, Dec. 1997.
- [8] O. Coudert, A. Narayan, "Subsystem: Logic Optimization", Tech. Report, Monterey Design Systems, Sept. 1998.
- [9] W. Donath, P. Kudva, L. Stok, P. Villarubia, L. Reddy, S. Sullivan, "Transformational Placement and Synthesis", Proc of *DATE'2000*, pp. 194–201, March 2000.
- [10] H. Eisenmann, F. M. Johannes, "Generic Global Placement and Floorplanning", in Proc. of *35th DAC*, pp. 269–274, June 1998.
- [11] J. P. Fishburn, A. E. Dunlop, "TILOS: a Posynomial Programming Approach to Transistor Sizing", *ICCAD'85*, pp. 326–328, Nov. 1985.
- [12] L. P. P. van Ginneken, "Buffer placement in distributed RC-tree networks for minimal Elmore delay", *ISCAS'90*, 1990.
- [13] J. Grodstein, E. Lehman, H. Harkness, B. Grundmann, Y. Watanabe, "A Delay Model for Logic Synthesis of Continuously-Sized Networks", *ICCAD'95*, pp. 264–271, Nov. 1995.
- [14] G. D. Hachtel, F. Somenzi, "Logic Synthesis and Verification Algorithms", Kluwer Academic Pub., 1996.
- [15] R. Haddad, L. P. P. van Ginneken, N. Shenoy, "Discrete drive selection for continuous sizing", *ICCD'97*, 1997.
- [16] S. Hojat, P. Villarubia, "An Integrated Placement and Synthesis Approach for Timing Closure of PowerPC Microprocessor", in *ICCD'97*, pp. 206–210, Sept. 1997.
- [17] B. Hoppe, G. Neuendorf, D. Schmitt-Landsiedel, "Optimization of High-Speed CMOS Logic Circuits with Analytical Models for Signal Delay, Chip Area and Dynamic Power Dissipation", in *IEEE Trans. on CAD*, 9-3, pp. 236–246, March 1990.
- [18] J. Hu, S. S. Sapatnekar, "Simultaneous Buffer Insertion and Non-Hanan Optimization for VLSI Interconnect under a Higher Order AWE Model", *ISPD'99*, 1999.
- [19] Y. Jiang, S. S. Sapatnekar, C. Bamji, J. Kim, "Interleaving Buffer Insertion and Transistor Sizing into a Single Optimization", *IEEE Trans. on VLSI*, Dec. 1998.
- [20] 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.
- [21] J. M. Kleinhans, G. Sigl, F. M. Johannes, K. J. Antreich, "GORDIAN: VLSI Placement by Quadratic Programming and Slicing Optimization", *IEEE Trans. on CAD*, 10, pp. 356–365, March 1991.

- [22] J. Lou, A. Salek, M. Pedram, “An Exact Solution to Simultaneous Technology Mapping and Linear Placement Problem”, *ICCAD’97*, pp. 671–675, Nov. 1997.
- [23] T. Okamoto, J. Cong, “Interconnect Layout Optimization by Simultaneous Steiner Tree Construction and Buffer Insertion”, *ISPD’96*, 1996.
- [24] M. Pedram, N. Bhat, “Layout driven technology mapping”, *28th DAC*, pp. 99–105, June 1991.
- [25] M. Pedram, N. Bhat, “Layout driven logic restructuring and decomposition”, *ICCAD’91*, Nov. 1991.
- [26] N. Shenoy, M. Iyer, R. Damiano, K. Harer, H.-K. Ma, P. Thilking, “A Robust Solution to the Timing Convergence Problem in High Performance Designs”, in Proc. of *ICCD’99*, pp. 250–257, Oct. 1999.
- [27] 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.
- [28] A. Salek, J. Lou, M. Pedram, “A simultaneous routing tree construction and fanout optimization algorithm”, *ICCAD’98*, pp. 625–630, Nov. 1998.
- [29] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, S. M. Kang, “An Exact Solution to the Transistor Sizing Problem for CMOS Circuits Using Convex Optimization”, *IEEE Trans. on CAD*, **12**-11, pp. 1621–1634, Dec. 1993.
- [30] C. Sechen, “VLSI Placement and Global Routing Using Simulated Annealing”, Kluwer Pub., Deventer, Netherlands, 1988.
- [31] 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.
- [32] D. Sylvester, K. Keutzer, “Getting to the Bottom of Deep Submicron”, in Proc. of *ICCAD’98*, pp. 203–211, Nov. 1998.
- [33] H. Zhou, M. Wong, I-M. Liu, A. Aziz, “Simultaneous Routing and Buffer Insertion with Restrictions on Buffer Locations”, *36th DAC*, June 1999.