Timing and Design Closure in Physical Design Flows

Olivier Coudert
Monterey Design Systems
894 Ross Drive, Suite 100
Sunnyvale, CA 94089-1443

Introduction

A physical design flow consists of producing a production-worthy layout from a gate-level netlist subject to a set of constraints. This paper focuses on the problems imposed by shrinking process technologies. It exposes the problems of timing closure, signal integrity, design variable dependencies, clock and power/ground routing, and design signoff. It also surveys some physical design flows, and outlines a refinement-based flow.

1. The Challenges of a Physical 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 or RTL). Place and route produces the layout from the gate-level netlist and technology files. 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 0.18µm and beyond, interconnect becomes a dominant factor and breaks the dichotomy between the logical and physical domains.

1.1 Timing Closure

We can no longer neglect the impact of the interconnect on timing: 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.

Using a statistical wire load model (WLM) based on pin count to estimate the interconnect delays does not work, because timing is determined by the maximum path delay, and although a WLM is 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. This means that 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 induces signal noise. Other physical aspects such as antenna effect, electromigration, self-heating and 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 since it is very costly and time consuming, if not impossible, to fix them during the final implementation stages.

1.3 Capacity

Designs keep getting bigger and more complex. Production designs today contain 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 the memory limit and computational resources of currently available CPUs (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 in several logical and physical blocks, with several independent design teams working in parallel at a different pace. Few very large chips are actually designed flat: the trend is towards more hierarchical designs.

Hierarchical flows pose new problems for physical design. For example, how to handle timing constraints between the chip level and the block level, how to verify the feasibility of the constraints at the chip level, how to capture the chip context when implementing a block, how to hierarchically verify the chip, etc. Floorplanning is still a difficult problem, especially when it must be done considering timing and congestion.

2. Survey of Current Flows

Some of the proposed solutions to solve the DSM challenges are as follows.

2.1 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-synthesized. 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, thus a different placement and routing, thus different wire loads, thus a different timing. There is no guarantee that this iterative process will converge.

2.2 Block-assembly flow

This flow is based on the idea that a statistical wireload model can still be used for small blocks of logic. Blocks smaller than

Proceedings of the International Symposium on Quality Electronic Design (ISQED'02)
0-7695-1561-4/02 $17.00 © 2002 IEEE

50k cells have been proposed [33]. 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, this 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, so that the intra-block delays are preserved, which can overconstrain placement. Third, it is virtually impossible to estimate the inter-block delays, since long interconnects depend on the relative placement of the blocks. Fourth, statistical wireload 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. Since congestion impacts wireload model predictability, the overall connectivity of the netlist must be considered, which is no longer a local block-level property.

2.3 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. Fixing the delay consists of fixing the gain. A fixed delay (i.e., fixed gain) is assigned on every logical stage so that timing constraints are met. Then these gains are preserved as the netlist is placed and routed.

Constant delay synthesis is attractive because of its elegance and simplicity, which enables fast synthesis. It has proven to be an efficient solution for RTL-to-gate synthesis. However, this elegance is obtained at the cost of ignoring the reality of physical design aspects. Delay models must be input slope dependent and distinguish between rising and falling signals. Such a simple model cannot capture the propagation of a waveform in a net. 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/routing/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 assumes convex input pin capacitance, which is often not true for real-life libraries.

2.4 Placement Aware Synthesis

This is a general trend, since synthesis needs placement information to estimate the timing [26, 32]. But 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 new timing problems. Making this flow converge is a major problem. Moreover, synthesis and placement working together is clearly not sufficient if it does not account for congestion and signal integrity issues, which require 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.

3. Refinement-based Flow

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, signal integrity, etc. The increasing dependency of the design quality on physical effects (crosstalk being the most obvious) has to be managed.

A variable of a design in process is always estimated with some uncertainty. For example, the timing of a design is fully known only if the 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 simply *useless* to do any kind of meaningful timing optimization beyond simple wirelength minimization.

Indeed, 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.

There is a point in this refinement process where timing can be predicted within 10% of the post-layout timing. This point is called a *physical prototype*, and can be used for early design signoff.

3.1 Placement

Quadratic placement is very fast and can handle large designs. However, it aims at minimizing the squares of the wirelengths, not the actual wirelengths. 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 problematic. Quadratic placement is not suitable for simultaneous optimization of other aspects of physical design, e.g., clock, crosstalk. However, because of its speed, it is often used to seed a more sophisticated placement method.

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. It is slower than quadratic placement, but gives significantly better results. However, there is a limit to what can be expressed with an analytical function, thus some costs are hard to capture, and the tuning of the "repulsive" terms can be difficult.

Simulated annealing [31] is based on move-and-cost. A move consists in 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 some probability that is slowly decreased to zero. Accepting cost degrading moves allows exploring larger solution spaces and avoiding local minima. Simulated annealing can produce globally optimum solutions, and it has a completely open cost function. 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 used for end case placement only (detailed 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 the same procedure is iterated. *Quadrisection* is the same process except that partitioning creates four partitions instead of two. Experience has shown that quadrisection placement 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 so 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, also called *bins*. When a local minimum is reached, the netlist is re-clustered, each partition is quadrisected, and the whole process is iterated. This cluster move-based method with an open cost function allows simultaneous optimization of all design aspects (including timing, congestion, wirelength, and crosstalk), 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 is a major advantage of refining the placement of objects in smaller and smaller bins: the placement is flexible enough to accommodate more or less disruptive netlist changes. One of the most important applications is to accommodate netlist changes performed by logic synthesis and optimization [16, 5]. Another crucial application, in the context of complex chip designs, 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, disconnect and reconnect a pin to 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. An ECO at the behavioral level is even more disruptive. Having several levels of placement resolution eases dramatically the integration of disruptive ECOs.

3.2 Logic Optimization

The gate-level netlist is initially synthesized with no placement information and, therefore, only with a crude estimation of delays. Timing becomes meaningful when enough placement information is available –unless one uses constant-delay synthesis with a high degree of confidence--. Only at that moment can the logic be revisited with a good understanding of timing, as well as congestion and power.

Physical logic 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 must interact with the placer and router.

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. The model 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 seen as a 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. Also an efficient incremental static timing analyzer is needed, 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.

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 should become 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 allowed. After detailed placement, only restricted sizing and non-disruptive buffering can be applied.

Common physical logic synthesis and optimization techniques are: (1) Load and driver strength optimization. This includes gate sizing, buffering, pins swapping, gate cloning. (2) Timing boundary shifting. This includes transparent latches (i.e., cycle stealing), useful skew, and retiming. (3) Resynthesis and technology remapping. (4) Redundancy-based optimization. (5) Area and/or power recovery. Some new logic transformations specific to physical design, such as synthesis of both the logic *and* the interconnect, synthesis for congestion [8] or logic optimization for signal integrity, are also emerging.

3.2.1 Sizing

The goal of gate sizing is to determine optimum sizes for the gates so that the circuit meets the design constraints (slope, setup, hold, max capacitance) with the least area/power cost. A larger gate will have a higher drive strength (lower resistance) and hence 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. Sizing requires a careful balancing of these conflicting effects, and an optimal solution will require the coordination of the correct sizes of all the gates along *and* off critical paths.

Analytical techniques exist for sizing in the continuous domain (e.g., linear programming [3], polynomial [11], convex programming [17, 30]), and various attempts have been made to use these results with discrete sizes in a cell library based design style [15, 2]. 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.

Analytical techniques cannot always capture the most accurate delay models (e.g., rising and falling delays). Also they 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.

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 and are not limited by assumptions needed for analytical techniques. Global search techniques 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.

3.2.2 Buffering

Buffering serves multiple functions: (1) Long wires result in signal attenuation (slope degradation). Buffers (or repeaters in this context) are used to restore signal levels. (2) A chain of one or more buffers can be used to increase the drive strength for a gate that is driving a large load. (3) 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 requirements. Algorithmic solutions are preferred that balance the attenuation, drive strength, and shielding requirements. This problem is hopelessly NP-hard. Even the problem of determining the best buffer tree for a given net, ignoring the placement of the driver and the sink pins, has been shown to be NP-hard [28]. 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 capture the resistive effects of DSM technologies. The former is considered to be more of a problem, since 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, 29], including adding additional degrees of freedom like wire and driver sizing. These techniques give better solutions than the fixed topology approach, but it is hard to say how optimal they are. Various constraints must be handled when routing and buffering the net, like routing blockages and restricted buffer locations due to placement keepouts [34, 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 this problem is very complex and far from being considered solved.

3.2.3 Resynthesis and 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 [14]. In the context of physical synthesis, the challenge is to work on sections small enough to not significantly disturb the placement, yet significant enough to improve the design. Another problem 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, which consists of resynthesizing some logic from scratch, is the most aggressive logic optimization, but it can significantly change the structure of the netlist. This also 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).

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, it is also desirable 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.

3.3 Clock Tree Synthesis

In the traditional flow, clock tree is a separate task. Typically, it is built once the locations of the sequential elements and gated clocks are known, i.e., after placement. But synthesizing clock trees after placement creates routing problems that occur too late to be fixed.

Clock tree synthesis must be part of the refinement process, because it contributes significantly to congestion and power dissipation, as well as timing using useful skew. Note that targeting a zero-skew clock tree has no justification but historical. First, skew can be used to optimize the timing. Second, a zero-skew clock tree means that all the sequential elements will toggle at the same time, which produces a peak power that is not desirable. After some level of quadrisection, the distribution of the sequential elements and gated clocks in the bins will not change substantially. 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. From that 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 avoids a congestion surprise at the end and 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, since the critical paths are continuously monitored.

Note that the scan chain 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.

3.4 Power/Ground Routing

The power/ground network can have a huge impact on congestion. For that reason, it must be put in place as early as possible so that its contribution to congestion is accounted for. 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 power/ground network. This allows an accurate assessment of the quality and integrity of the power routing because the power rail currents will not change much as the placement is further refined.

Power consumption depends on net capacitances and toggle rates (number of transitions on a net per unit of time). The switching activity can obtained by an external simulation (e.g., VCD file), or be computed on-the-fly using simulation (slow, but accurate) or by probabilistic analysis (fast, but sometime misleading). Knowing the distribution of the current sources in the bins, one can extract a power network that is simulated to produces an IR drop map. This helps in adjusting the power grid since placement is still flexible enough at that stage to accommodate adding and/or widening power stripes.

3.5 Crosstalk

Fixing crosstalk post-layout 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., setting 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 irresolvable congestion problems.

There have been 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 identify 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. This requires the detailed router to have gridless capabilities with variable width and variable spacing so that crosstalk issues can be addressed effectively.

3.6 IR Drop

IR Drop is the problem of voltage drop on the power and ground due to high current flowing through the power/ground resistive network. When the voltage drop (or rise in the ground net) becomes excessive, this causes delays in gates, which can produce timing violations. If severe enough, IR drop may also cause unreliable operation because of reduced noise margins.

IR drop becomes more critical for DSM designs because: (1) there are more devices, thus there is a higher current demand; (2) the wire and contact/via resistance of the supply network increases because of narrower wires and fewer contacts/vias; (3) the supply voltage is decreased to 1.5 volts and below; (4) noise margins scale as well. As an example, the delay increase produced by a 0.15V drop through a typical inverter at 0.18μm is 10%, and is more than 60% for a 0.5V drop.

For many designs today, 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.

An accurate IR drop analysis is done with a transistor level simulation that computes the dynamic current flows. This is a costly process that takes far too long to perform for a complete chip. Fortunately, the simulation can be done at the cell level using cell-level power models, an RC model for the interconnect, and data on switching direction and frequency. This produces average current per cell, from which the average voltage drop can be approximated using a resistance model of the power and ground network. Although less accurate, this method can identify regions with voltage drops high enough to be significant.

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 that 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.

3.7 Electromigration

When there is too much current density through the interconnect for an extended period of time, its resistance increases. This results in self-heating and metal disintegration, which eventually causes an open or short in the circuit. This effect is called electromigration (EM). It becomes more severe with the advent of DSM. As circuits 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. As explained above, however, over-designing the power network is costly in terms of routing resources. 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 that provides sufficient interconnect width without excessive over-design is necessary.

Tools that compute current densities from the final layout are often used to analyze EM problems, which are then fixed manually by widening the problem nets. However, this requires considerable expertise and time, 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, the current flow in these nets can be calculated, which, together with the switching

activity of the nets, enables electromigration analysis. The interconnect width and via count needed to support the current are identified at each level of quadrisection. This drives the routing resources allocated by the global router. Consequently, the detailed router will be able to satisfy the electromigration routing requirements together with other requirements.

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

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 flow based on placement and routing refinement (including clock tree, power routing, and scan chain) with an open cost function, together with physical logic synthesis and optimization, can meet these challenges. It enables early estimation of congestion, timing, and physical effects at a point in the flow when the design has enough flexibility to accommodate perturbations produced by the optimization procedures. The physical prototype produced by the refinement-based flow restores the design signoff solution lost in the mid-90's.

The interconnect centric DSM era raises new issues, where placement, routing, and logic optimization are tightly interdependent. Further, the introduction of hierarchy to handle multi-million gate designs requires new full-chip design system.

References

[1] R. Arunachalam, K. Rajagopal, L. Pileggi, "TACO: Timing Analysis with Coupling", Proc. of ICCAD'2000, Nov. 2000.

[2] F. Beeftink, P. Kudva, D. Kung, L. Stok, "Gate-Size Selection for Standard Cell Libraries", ICCAD'98, Nov. 1998.

[3] M. Berkelaar, J. Jess, "Gate Sizing in MOS Digital Circuits with Linear Programming", Proc. of EDAC'90, 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.

[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, pp. 465-472, Dec. 1997.

[8] O. Coudert, A. Narayan, "Subsystem: Logic Optimization", Tech. Report, Monterey Design Systems, Sept. 1998.

[10] H. Eisenmann, F. M. Johannes, "Generic Global Placement and Floorplanning", in Proc. of 35th DAC, 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. 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, Nov. 1995.

[14] G. D. Hachtel, F. Somenzi, "Logic Synthesis and Verification Algorithms", Kluwer Academic Pub., 1996.

[15] R. Haddad, L. P. 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 Microporcessor", 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, 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.

[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, June 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", ICCD'99, 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] H. J. Touati, "Performance Oriented Technology Mapping", UCB Ph.D. Thesis, 1990.

[29] A. Salek, J. Lou, M. Pedram, " A simultaneous routing tree construction and fanout optimization algorithm", ICCAD'98, 1998.

[30] 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, pp. 1621-1634, Dec. 1993.

[31] C. Sechen, "VLSI Placement and Global Routing Using Simulated Annealing", Kluwer Pub., Deventer, Netherlands, 1988.

[32] 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.

[33] D. Sylvester, K. Keutzer, "Getting to the Bottom of Deep Submicron", in Proc. of ICCAD'98, Nov. 1998.

[34] H. Zhou, M. Wong, I-M. Liu, A. Aziz, "Simultaneous Routing and Buffer Insertion with Restrictions on Buffer Locations", 36th DAC, June 1999.