8.6 Performance-Driven Design Flow

The previous sections have presented a broad range of algorithms and techniques that can be used to improve the timing of modern digital circuit designs. This section takes the next step, by combining these optimizations in a consistent *performance-driven physical design flow*, which seeks to satisfy timing constraints, i.e., "close on timing." Due to the nature of performance optimizations, the order of optimizations is important, and their interactions with conventional layout techniques are subject to a number of subtle limitations. Evaluation steps, particularly STA, must be invoked several times, and some optimizations, such as buffering, must often be redone multiple times to facilitate a more accurate evaluation.

Baseline Physical Design Flow Recall that a typical design flow starts with *chip planning* (Chap. 3), which includes *I/O placement*, *floorplanning* (Fig. 8.22), and *power planning*. *Trial synthesis* provides the floorplanner with an estimate of the total area needed by modules. Besides logic area, additional whitespace must be allocated to account for buffers, routability, and gate sizing.

Then, *logic synthesis* and *technology mapping* produce a gate-level netlist from a high-level specification, which is tailored to a specific technology library. Next, *global placement* assigns locations to each movable object (Chap. 4). As illustrated in Fig. 8.23, most of the cells are clustered in highly concentrated regions (colored black). As the iterations progress, the cells are gradually spread across the chip, such that they overlap less and less (colored light gray).

These locations, however, do not have to be aligned with cell rows or sites, and slight cell overlap is allowed. To ensure that the overlap is small, it is common to (1) establish a uniform grid, (2) compute the total area of objects in each grid square, and (3) limit this total by the area available in the square.

After global placement, the sequential elements are legalized. Once the locations of sequential elements are known, a *clock network* (Chap. 7) is generated. ASICs, SoCs, and low-power (mobile) CPUs commonly use clock trees (Fig. 8.24), while high-performance microprocessors incorporate structured and hand-optimized clock distribution networks that may combine trees and meshes [25, 26].

A. B. Kahng, J. Lienig, I. L. Markov, J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, Springer 2022, https://doi.org/10.1007/978-3-030-96415-3_8

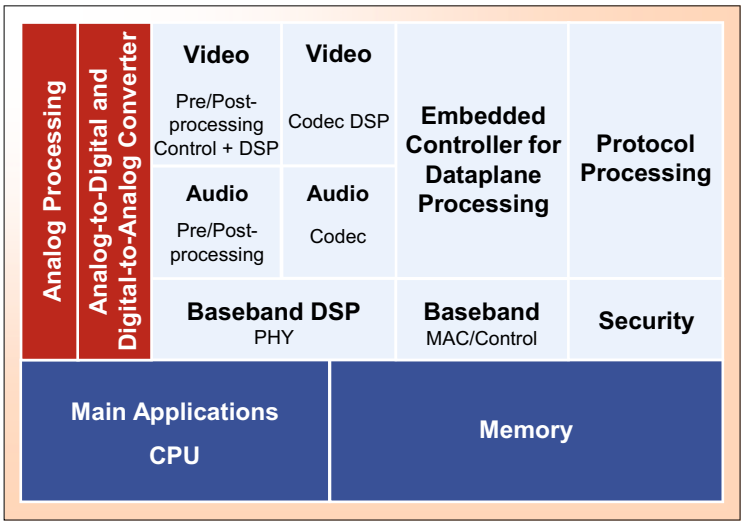

Fig. 8.22 An example floorplan of a system-on-chip (SoC) design. Each major component is given dimensions based on area estimates. The audio and video components are adjacent to each other, given that their connections to other blocks and their performance constraints are similar

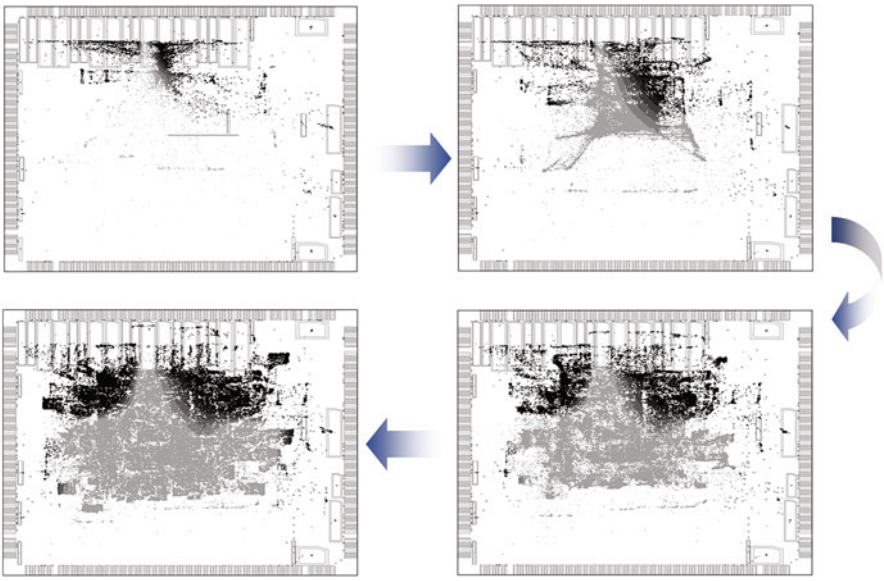

Fig. 8.23 The progression of cell spreading during global placement in a large, flat (non-floorplanned) ASIC design with fixed macro blocks. Darker shades indicate greater cell overlap while lighter shades indicate smaller cell overlap

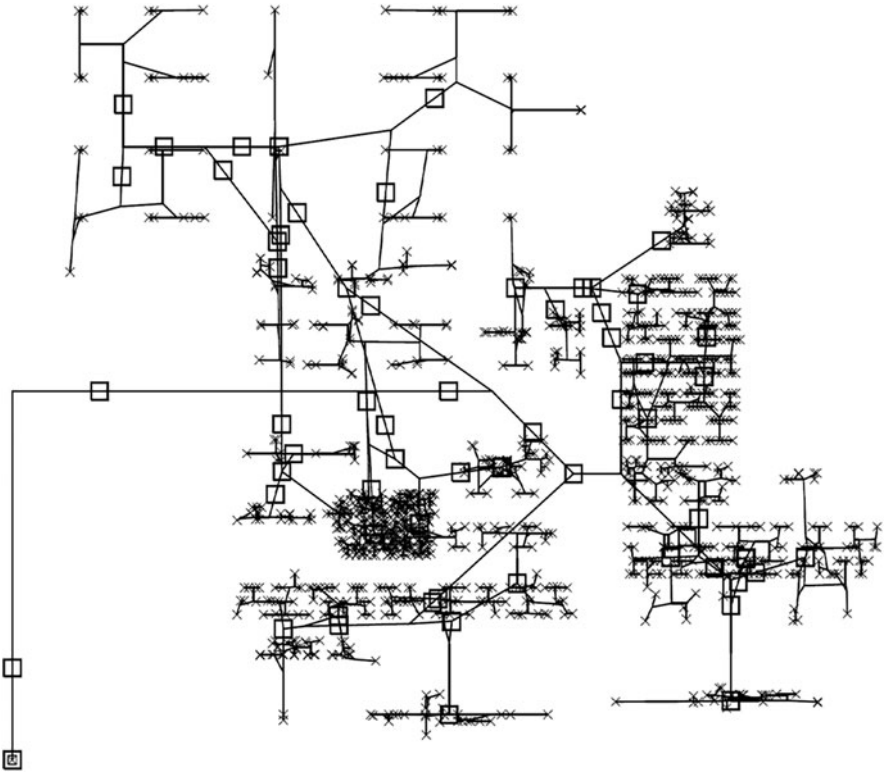

Fig. 8.24 Buffered clock tree in a small CPU design. The clock source is in the lower left corner. Crosses (×) indicate sinks, and boxes (□) indicate buffers. Each diagonal segment represents a horizontal plus a vertical wire (*L*-shape), the choice of which can be based on routing congestion

The locations of globally placed cells are first temporarily rounded to a uniform grid, and then these rounded locations are connected during *global routing* (Chap. 5) and *layer assignment*, where each route is assigned to a specific metal layer. The routes indicate areas of wiring congestion (Fig. 8.25). This information is used to guide *congestion-driven detailed placement* and *legalization* of combinational elements (Chap. 4).

While detailed placement conventionally comes before global routing, the reverse order can reduce overall congestion and wirelength [23]. Note that EDA flows typically require a legal placement before global routing. In this case, legalization will be performed after global placement. The global routes of signal nets are then assigned to physical routing tracks during *detailed routing* (Chap. 6).

The layout generated during the place-and-route stage is subjected to *reliability*, *manufacturability*, and *electrical verification*. A subsequent *layout post-processing* step inserts amendments and additions to the chip layout and might apply resolution enhancement techniques (RET) [34]. Afterwards, each standard cell and each route are represented by collections of rectangles in a format suitable for generating optical

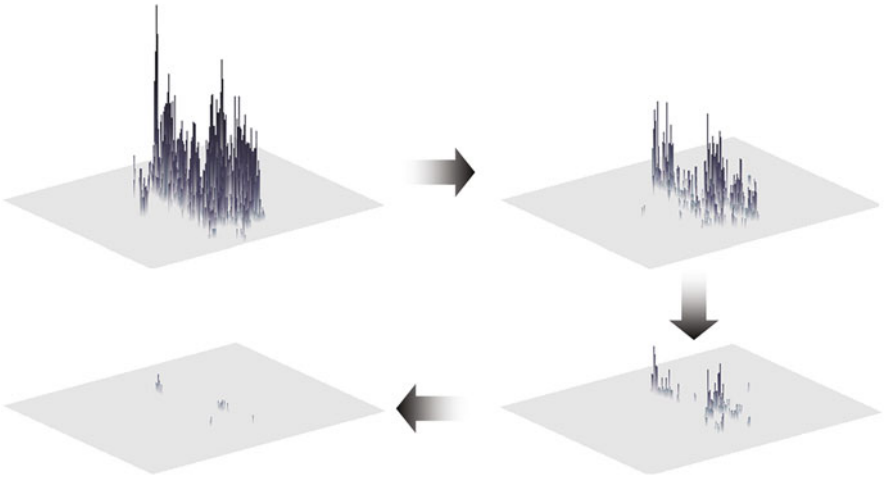

Fig. 8.25 Progression of congestion maps through iterations of global routing. The light-colored areas are those that do not have congestion; dark-colored peaks indicate congested regions. Initially, several dense clusters of wires create edges that are far over capacity. After iterations of rip-up and reroute, the route topologies are changed, alleviating the most congested areas. Though more regions can become congested, the maximum congestion is reduced

lithography masks for chip fabrication. With this, the physical layout is converted into data for mask production (*mask generation*).

This baseline PD flow is illustrated in Fig. 8.26 with white boxes. Extending the baseline design flow, contemporary industrial flows are typically built around static timing analysis and seek to minimize the amount of change required to close on timing. Some flows start timing-driven optimizations as early as the chip planning stage, while others do not account for timing until detailed placement to ensure accuracy of timing results. This section discusses the timing-driven flow illustrated in Fig. 8.26 with colored boxes. Advanced methods for physical synthesis are found in [3].

Chip Planning and Logic Design Starting with a high-level design, performance-driven chip planning generates the I/O placement of the pins and rectangular blocks for each circuit module while accounting for block-level timing, and the power supply network. Then, logic synthesis and technology mapping produce a netlist based on delay budgets.

Performance-Driven Chip Planning Once the locations and shapes of the blocks are determined, global routes are generated for each top-level net, and buffers are inserted to better estimate timing [2]. Since chip planning occurs before global placement or global routing, there is no detailed knowledge of where the logic cells will be placed within each block or how they will be connected. Therefore, buffer insertion makes optimistic assumptions.

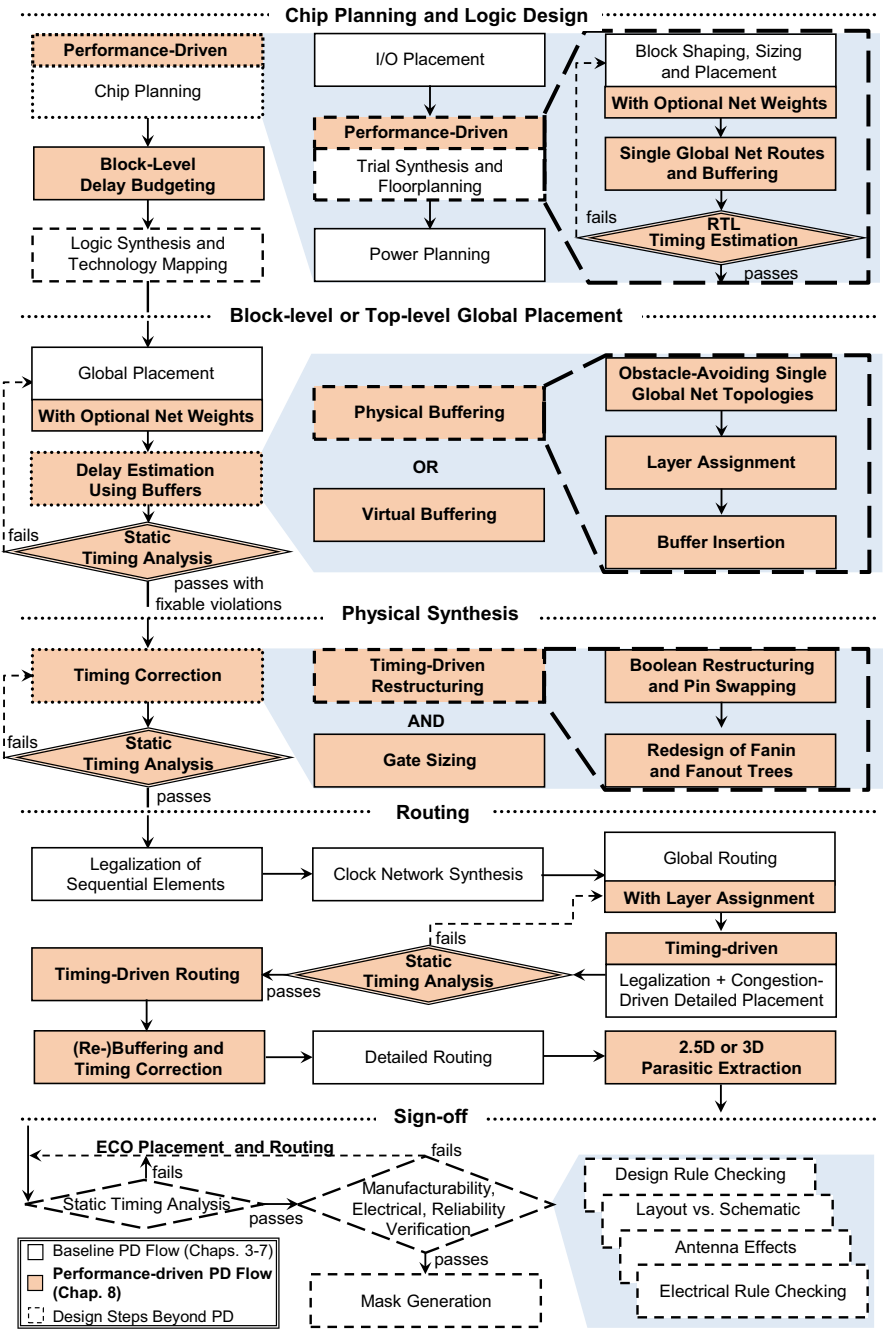

Fig. 8.26 Integrating optimizations covered in Chaps. 3–8 into a performance-driven design flow. Some tools bundle several optimization steps, which changes the appearance of the flow to users and often alters the user interface. Alternatives to this flow are discussed in this section

After buffering, STA checks the design for timing errors. If there are a sufficient number of violations, then the logic blocks must be re-floorplanned. In practice, modifications to existing floorplans to meet timing are performed by experienced designers with little to no automation. Once the design has satisfied or mostly met timing constraints, the I/O pins can be placed, and power (VDD) and ground (GND) supply rails can be routed around floorplan blocks.

Timing Budgeting After performance-driven floorplanning, delay budgeting sets upper bounds on *setup* (long path) timing for each block. These constraints guide logic synthesis and technology mapping to produce a performance-optimized gate-level netlist, using standard cells from a given library.

Block-level or Top-level Global Placement Starting at global placement, timing-driven optimizations can be performed at the *block level*, where each individual block is optimized, or *top level*, where transformations are global, i.e., cross-block boundaries, and all movable objects are optimized.[5] Block-level approaches are useful for designs that have many macro blocks or *intellectual properties* (*IPs*) that have already been optimized and have specific shapes and sizes. Top-level approaches are useful for designs that have more freedom or do not reuse previously designed logic; a hierarchical methodology offers more parallelism and is more common for large design teams.

Buffer Insertion To better estimate and improve timing, buffers are inserted to break any extremely long or high fanout nets (Sect. 8.5.2). This can be done either *physically*, where buffers are directly added to the placement, or *virtually*, where the impact of buffering is included in delay models, but the netlist is not modified.

Physical Buffering Physical buffering [1] performs the full process of buffer insertion by (1) generating obstacle-avoiding global net topologies for each net, (2) estimating which metal layers the route uses, and (3) actually inserting buffers (Fig. 8.27).

Virtual buffering [19], on the other hand, estimates the delay by modeling every pin-to-pin connection as an optimally buffered line with linear delay as

$$t_{LD}(net) = L(net) \cdot \left(R(B) \cdot C(w) + R(w) \cdot C(B) + \sqrt{2 \cdot R(B) \cdot C(B) \cdot R(w) \cdot C(w)} \right)$$

where *net* is the net, $L(net)$ is the total length of *net*, $R(B)$ and $C(B)$ are the respective intrinsic resistance and capacitance of the buffer, and $R(w)$ and $C(w)$ are the respective wire resistance and capacitance. Though no buffers are added to the netlist, they are assumed for timing purposes. When timing information becomes

[5] In hierarchical design flows, different designers concurrently perform top-level placement and block-level placement.

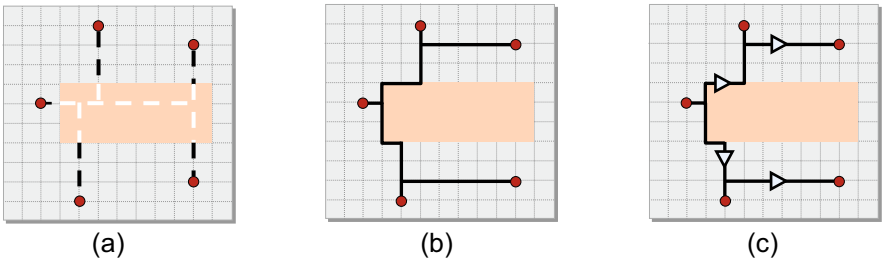

Fig. 8.27 Physical buffering for timing estimation. (**a**) A five-pin net is routed with a minimum Steiner tree topology that does not avoid a routing obstacle (shown in orange). (**b**) The net routed with an obstacle-avoiding Steiner tree topology. (**c**) The buffered topology offers a relatively accurate delay estimation

more accurate, subsequent re-buffering steps often remove any existing buffers and reinsert them from scratch. In this context, virtual buffering saves effort while preserving the accuracy of timing analysis. Physical buffering can avoid unnecessary upsizing of drivers and is more accurate than virtual buffering, but is also more time-consuming.

Once buffering is complete, the design is checked for timing violations using static timing analysis (STA) (Sect. 8.2.1). Unless timing is met, the design returns to buffering, global placement, or, in some cases, logic synthesis. When timing constraints are mostly met, the design moves on to timing correction, which includes gate sizing (Sect. 8.5.1) and timing-driven netlist restructuring (Sect. 8.5.3). Subsequently, another timing check is performed using STA.

Physical Synthesis After buffer insertion, physical synthesis applies several timing correction techniques (Sect. 8.5) such as operations that modify the pin ordering or the netlist at the gate level, to improve delay on critical paths.

Timing Correction Methods such as *gate sizing* increase (decrease) the size of a physical gate to speed up (slow down) the circuit. Other techniques such as *redesign of fan-in and fanout trees*, *cloning*, and *pin swapping* reduce timing by rebalancing existing logic to reduce load capacitance for timing-critical nets. Transformations such as *gate decomposition* and *Boolean restructuring* modify logic locally to improve timing by merging or splitting logic nodes from different signals. After physical synthesis, another timing check is performed. If it fails, another pass of timing correction attempts to fix timing violations.

Routing After physical synthesis, all combinational and sequential elements in the design are connected during global and clock routing, respectively. First, the sequential elements of the design, e.g., flip-flop and latches, are legalized (Chap. 4, Sect. 4.4). Then, clock network synthesis generates the clock tree or mesh to connect all sequential elements to the clock source. Modern clock networks

require a number of large clock buffers[6]; performing clock network design *before* detailed placement allows these buffers to be placed appropriately. Given the clock network, the design can be checked for *hold-time (short path) constraints*, since the clock skews are now known, whereas only *setup (long path) constraints* could be checked before.

Layer Assignment After clock network synthesis, global routing assigns global route topologies to connect the combinational elements. Then, layer assignment matches each global route to a specific metal layer. This step improves the accuracy of delay estimation because it allows the use of appropriate *resistance-capacitance* (*RC*) parasitics for each net. Note that clock routing is performed before signal net routing when the two share the same metal layers—clock routes take precedence and should not detour around signal nets.

Timing-Driven Detailed Placement The results of global routing and layer assignment provide accurate estimates of wire congestion, which is then used by a congestion-driven detailed placer [8, 30]. The cells are (1) spread to remove overlap among objects and decrease routing congestion, (2) snapped to standard cell rows and legal cell sites, and then (3) optimized by swaps, shifts, and other local changes. To incorporate timing optimizations, *either* perform (1) non-timing-driven legalization followed by timing-driven detailed placement *or* (2) perform timing-driven legalization followed by non-timing-driven detailed placement. After detailed placement, another timing check is performed. If timing fails, the design could be globally rerouted or, in severe cases, globally replaced.

To give higher priority to the clock network, the sequential elements can be legalized first, and then followed by global and detailed routing. With this approach, signal nets must route around the clock network. This is advantageous for large-scale designs, as clock trees are increasingly becoming a performance bottleneck. A variant flow, such as the industrial flow described in [23], first fully legalizes the locations of all cells, and then performs detailed placement to recover wirelength.

Another variant performs detailed placement before clock network synthesis, and then is followed by legalization and several optimization steps.[7] After the clock network has been synthesized, another pass of setup optimization is performed. Hold violations may be addressed at this time or, optionally, after routing and initial STA.

Timing-Driven Routing After detailed placement, clock network synthesis, and post-clock network synthesis optimization, the *timing-driven routing* phase aims to fix the remaining timing violations. Algorithms discussed in Sect. 8.4 include generating *minimum-cost, minimum-radius trees* for critical nets (Sects. 8.4.1, 8.4.2), and *minimizing the source-to-sink delay* of critical sinks (Sect. 8.4.3).

[6] These buffers are legalized immediately when added to the clock network.

[7] These include post-clock network synthesis optimizations, post-global routing optimizations, and post-detailed routing optimizations.

If there are still outstanding timing violations, further optimizations such as re-buffering and late timing corrections are applied. An alternative is to have designers manually tune or fix the design by relaxing some design constraints, using additional logic libraries, or exploiting design structure neglected by automated tools. After this time-consuming process, another timing check is performed. If timing is met, then the design is sent to detailed routing, where each signal net is assigned to specific routing tracks. Typically, incremental STA-driven *Engineering Change Orders* (*ECOs*) are applied to fix timing violations after detailed placement; this is followed by ECO placement and routing. Then, *2.5D* or *3D parasitic extraction* determines the electromagnetic impact on timing based on the routes' shapes and lengths, and other technology-dependent parameters.

Sign-off The last few steps of the design flow validate the layout and timing, as well as fix any outstanding errors. They also ensure manufacturability and finally convert a physical layout into data for mask production.

If a timing check fails, ECO minimally modifies the placement and routing such that the violation is fixed and no new errors are introduced. Since the changes made are very local, the algorithms for ECO placement and ECO routing differ from the traditional place and route techniques discussed in Chaps. 4–7.

After completing timing closure, *manufacturability*, *reliability*, and *electrical verification* ensure that the design can be successfully fabricated and will function correctly under various environmental conditions. The four main components are equally important and can be performed in parallel to improve runtime.

- *Design rule checking* (*DRC*) ensures that the placed-and-routed layout meets all technology-specified design rules, e.g., minimum wire spacing and width.
- *Layout vs. schematic* (*LVS*) checking ensures that the placed-and-routed layout matches the original netlist.
- *Antenna checks* seek to detect undesirable *antenna effects*, which may damage a transistor during plasma etching steps of manufacturing by collecting excess charge on metal wires that are connected to transistor gates [34]. This can occur when a route consists of multiple metal layers and a charge is induced on a metal layer during fabrication.
- *Electric rule checking* (*ERC*) finds all potentially dangerous electric connections, such as floating inputs and shorted outputs.

Following the abovementioned verification tasks, further *layout post-processing* steps are performed [34]. Here, amendments and additions to the chip layout data are implemented, such as test patterns and alignment marks [34]. Furthermore, *resolution enhancement techniques* (*RET*) might be applied to counter manufacturing and optical effects caused by extremely small feature sizes.

Finally, *optical lithography masks* are generated for manufacturing.