Combating congestion in high-performance, low-cost systems on chip

Abhishek Roy - February 23, 2010

The semiconductor chip market has become increasingly competitive in modern times. The customers demand extensive features for diverse applications, and hence, to pacify their needs most semiconductor players are designing SOCs (systems on chip) deploying increasingly complex architectures. An example can be multicore devices. Another example can be that to achieve greater test coverage through numerous test patterns, compression logic is incorporated in most SOCs for DFT (design-for-test) requirements. Also, the customers want their applications to run at higher speeds. As a result, the operating frequency for an SOC is higher than ever, on the order of several gigahertz.

To amass greater revenue than others, most semiconductor giants are cutting down on manufacturing costs by conserving on the die area. Fewer metal layers for interconnects saves additional mask-generation expenses. It also saves the manufacturing time in a semiconductor fab, thus shrinking the time to market for a device, which gives a significant competitive edge over other competitors.

Diminishing die sizes, fewer interconnect layers, and packing more and more IP blocks for high-end applications all pose greater design challenges and result in a very common phenomenon in physical design called congestion. A very high degree of congestion may render the design unroutable and, hence, inhibit design convergence. Congestion issues create one of the most difficult bottlenecks in design closure and should be spotted and tackled during the early phase of the physical design cycle.

This article sums up the most important among the many reasons for design congestion. It also explains why some of the most trivial-looking requirements can add to congestion if not taken care of. This article also presents some of the classic and some customized solutions for mitigating congestion in sub-90-nm SOCs. Finally, this article encompasses some important dos and don’ts for relieving congestion and ensuring a smooth transition through the physical design cycle.

Reasons for congestion

There can be multitude of reasons for congestion, with some reasons having a direct and others having an indirect impact. Let’s examine some.
Traditionally, congestion comes near the corners of hard macros such as memory blocks (RAM and flash, for example), voltage regulators, ADCs, and other analog blocks. In most cases, near the corners of such macros the signal nets change direction from horizontal to vertical, or vice versa, unless over-the-block routing is permitted. But over-block routing is generally prohibited, as the hardened macros would have their own inherent routing or may have special requirements, such as preventing crosstalk for proper device operation. So if standard cells are placed near the corners of such macros, they pose an obstacle to this change of net direction as additional routing resources are required to connect to such standard cells. This can cause an increase in routing congestion.

In the figure, you can see that the signal nets are changing direction, from horizontal (green) to vertical (red), at the corner of a hard macro (blue). So the density of nets increases if standard cells are sitting near the corner.

Channels between macros
Here's another example. The channel space between hard macros is most prone to routing congestion. This can happen even if standard cells are not placed within the channel. If there is congestion in the channel, it means that there is a mistake in estimation while floor-planning. The calculation of channel space is based on the number of nets passing through the channel and the appropriate number of horizontal/vertical routing resources (tracks) available, depending on the shape of the channel and direction of the nets passing through the channel. You can see in Figure 2 that a large number of horizontal routes (in blue and green) are passing through the channel, making it densely packed and leaving no room for additional routes.

**Timing-critical zones**

Also, certain areas of the design can be timing-critical. For example, the paths through the address pins of a flash memory block in an SOC are among the most setup-timing-critical, as such pins have large setup time requirements and flash IPs have huge access time. Additionally, flash operates on the system clock frequency. Ironically, this very tight timing constraint will result in a significant increase in standard-cell utilization in those areas, primarily because of logic getting optimized—cells getting upsized and buffers being added to improve transition, for example—and cells being placed close to each other for lesser net delays. Naturally, a large number of routes will be used for tapping these cells in a localized area, resulting in increased routing congestion.

Another example occurs in designs where the clock tree is not robust—for instance, when the clock skew is not optimized and balanced. Hold violations of high magnitude will occur. To resolve these violations, a large number of buffers may be added in data paths. These buffers are logically
redundant and may not be necessary when the device is operating in a different mode. These buffers are therefore uncalled for, as they also consume chip power. They also increase the utilization in localized areas, resulting in more routing congestion.

**Power issues get in the way**

If standard cells are placed beneath power straps, the routes are unable to access the pins of these standard cells, creating congestion ( ). From the illustration you can see that for tapping the pins in metal 1 (blue), you can’t use routes in metal 4 (yellow), as the metal 3 power route (in green) is blocking the pins. Hence, you can’t make connection from metal 4 to metal 1. So if you’ve consumed a number of routes in other metal layers, it will be difficult to connect to these pins without design-rule violations, such as shorts. Hence, this will result in routing congestion.

Also, the power grid may be designed such that the tappings from the topmost power strap to the follow-pins (the straps that supply power to standard cells) are done with stacked vias. These stacked vias are generally bigger in size than normal vias and consume a lot of routing tracks that could otherwise have been used for signal routing. For an overcongested design, this can have serious implications.
Another important point is that floor-planning issues and improper design of floorplan objects can increase congestion. With increased focus on low-power designs, most SOCs are based on low-power architectures. So when you design a power-domain fence in the floorplan, if the fence is not of appropriate dimensions the power domain may have very high utilization. This discrepancy can cause congestion, especially at the boundary of the fence ( ).

You can see from the diagram that the standard cells are densely packed at the power-domain boundary. A more appropriate shape and size for the power domain would help.
In certain areas within the sea-of-gates area, the localized pin density is quite high. This may be because the standard cells are getting closely placed together, or because the cells in a region have a large number of input pins and hence a large number of connections to these pins are necessary. As a large number of wire segments pass through a localized area, the routing congestion is increased many-fold within that region.

). The highlighted regions in the figure show that the use of cells with relatively higher pin density has created localized hot spots where such cells are sitting close together. This has resulted in regions with high pin density and a large number of nets passing through a localized area, resulting in congestion.

**Tackling congestion**

Analyzing and detecting congestion-prone regions of the design early in the physical design cycle is of utmost importance for faster design closure. Also, you should take care at every function to ensure that you maintain congestion control. Broadly speaking, the congestion can be mitigated by incorporating the following strategies.

For relieving congestion near the corners of hard macros, put hard placement blockages in the form of jogs (
By doing so, fewer cells will be placed around the corners; hence, some tracks are vacated for nets that change direction around the corners. You can finalize the size and shape of the jog after place-and-route iterations to predict the actual number of tracks that are required for the change of direction of the nets around the corners. Also, you should remove the follow-pins beneath these jogs. As no standard cell will be placed at these locations, there is no need to supply power to these unwanted straps. Some additional routing tracks can be saved by doing this.

Taming timing-critical regions

There are some regions within the sea of gates that are timing-critical. Examples of such paths can be from memories such as flash and RAM to core. As these areas are timing-critical, the standard cells placed within the modules may be placed close to each other. Naturally, an increased number of nets will be passing through these regions, creating congestion. To neutralize such congestion issues, you can add density screens in selected regions. These screens will ensure that the standard-cell density will not increase beyond a certain threshold.

But addition of density screens has a disadvantage. It may affect the timing of those critical paths in
order to meet congestion requirements. Apart from density screens, you can use module padding in select timing-critical modules. Module padding means intentionally inflating the effective area of the module so that the cells are more spread out throughout the module boundary.

Instance- or cell-specific padding is another solution ( ). By padding, we mean assigning a clearance limit to cells present in such heavily congested areas. These limits will cause neighboring cells to be spaced apart, maintaining that clearance value. Hence, the cells won’t be packed closely together and there would be lesser congestion due to cell placement. Instead of adding a density screen over a part of the design, or the entire module, you can give padding to certain instances or cells. In this way you target selected congestion-prone areas. You should identify these instances or cells after repetitive place-and-route iterations, so you can be sure that in every iteration, the selected cells are the culprit for congestion.

As evident from the red arrows in the highlighted region in the figure, the instance padding creates more space between the cells, accommodating more room for the routes. Also, if the region is more timing-critical and you have some leeway in leakage power, then you should opt for standard $V_{th}$ cells. Having lesser threshold voltage, these cells would most definitely have lesser propagation delays and, hence, would adhere to the high-frequency requirements. This would also mean you will get fewer buffers for improving transition and fixing design rule violations, reclaiming more area and helping to mitigate congestion.

**Power-routing fixes**
Ideally, you should not place standard cells beneath the power straps that supply power to the core. In certain cases ( ) where the strap supplying power is routed in metal 2 (red) while the standard cells have pins in metal 1, there is no way these pins can be tapped by routes. This conflict will result in pin access violations and opens—unconnected wires—causing LVS issues. The only way to prevent such issues is by blocking the region where such power straps are routed from standard-cell placement.

Another scenario where inefficient power routing can contribute to routing congestion is depicted in a) Stacked via approach and b) Continuous strap approach.
where stacked vias from a top power strap are brought down to the bottom metal strap supplying power to standard cell pins. As a result, a number of tracks are blocked for signal nets. You can avoid this waste of tracks by tapping power from top metal to bottom power strap only at the ends. If you see IR-drop issues, you can make the power grid continuous and uniform within the core area ( ), without placing stacked vias that unnecessarily consume all routing tracks starting from the top and ending at the bottom interconnect layer.

**Iterative floor planning**

To eliminate congestion at an early stage, substantial iterations between floor-planning and place+route are essential. As explained previously, inefficient macro placement can lead to
formation of congested channels within the core region. While in complex designs the formation of such channels is inevitable, you should take care in estimating the width of the channel. Do a rough calculation based on the number of signal nets passing through the channel and the amount of routing resources available. Routing resources include all routing tracks available in all layers within that channel space. As shown in

![Image showing channel width calculation](image)

the number of horizontal wires in metal 1 (blue) and metal 3 (green) are just about sufficient for the channel. So calculation needs to be done accurately to determine the space between hard macros.

Orientation of macros is very important. The pins of a hard macro should be easily accessible by the core logic. In fact, the pins should ideally be facing the standard-cell logic. If you don't take care of this issue during floor-planning, logical nets will travel around the macro—as routing over-macro is generally not allowed—resulting in long nets. These long nets not only contribute to congestion but also are prone to design-rule and timing violations.

In multipower domain designs, there are hard constraints in the form of power domain fences that restrict certain modules from being placed at particular locations. The coordinates and size of these floorplan objects, such as fences and guides, need to be accurately determined. The size should be such that the utilization within these fences/guides is optimum. If the utilization is too high, then the region may become prone to congestion ( ); if it is too low, then a lot of routing resources get wasted. Also, frequently the density at the boundaries of these fences is substantially high. To reduce congestion in these regions, you can apply cell-instance padding on selected cells at the boundaries.

**Clock trees and congestion**
Controlling congestion while building clock trees and optimizing skew is essential. Many a time while optimizing for skew, clock-gating cells and respective registers tend to sit very near to each other. In sub-90-nm technology, it is recommended to use double spacing for clock wires to reduce coupling and, hence, susceptibility to signal-integrity issues. Also, double width is used for resolving signal EM issues, particularly if the chip operating frequency is high. This can result in selected congestion hot spots in the design as a number of tracks are consumed to adhere to this special routing rule.

One way of resolving the issue is to space the affected standard cells apart by applying cell-instance padding. Another method is to route the clock wires of leaf cells (the registers) with single spacing and single width or double spacing, single width as applicable. Please note that only the last level—the driver to the load pins of these registers—should follow this rule. The rest of the net from the clock source should follow the original double-width, double-spacing rule. This should take care of routing congestion and cause minimal impact to noise and signal electromigration.

You can see that after implementing the proposed methodology, the congestion due to clock nets is hugely reduced.
In some selected region, downsizing standard cells to reclaim area for resolving congestion issues may not solve the problem. In fact, it may worsen the situation. The reason for this problem is that the effective pin density of the region is alarmingly high. Consider the following situation ( ). Most cells are small in size, but the number of pins in the highlighted area is high. Naturally, a large number of routes need to tap these pins, creating more wires in a small region.

One solution can be to space the cells apart by adding instance or cell padding. Another solution can be to avoid cells with multiple input pins, such as an eight-input AND gate or a nine-input NOR gate. The logic can be implemented using simpler cells, such as two four-input AND gates, which will have a lower number of pins. Thus, using such cells will lessen the probability of congestion due to high pin density. Also, it may help with better timing optimization.

Familiarity with the technology LEF file is of utmost importance. You should keep the routing pitch and interconnect spacing requirements in mind while estimating the shape of the floorplan during the power grid design and also during congestion analysis. Appropriate distribution of available interconnect layers for power distribution and signal routing is a necessity. For example, in a four-metal 90-nm process, it is imperative that most of the power grid is made in the upper metal layers so that signal-routing resources are fully available in the lower layers. And remember in which layers the standard cells' inherent geometries are present.

**Congestion-aware logical synthesis and STA**

Precautions need to be taken during logic synthesis and during STA (static timing analysis) to avoid scenarios where a congested design becomes inevitable. During synthesis, every piece of redundant logic if present and preserved should be questioned. Isolation-cell and clock-gating insertion should be as optimal as possible. You should apply optimization techniques such as cloning-decloning,
netlist restructuring, and fanout fixing with maximum possible effort.

During STA, take care that only necessary paths are being optimized. There may be cases that paths operating at lower frequency are being optimized at high frequencies, resulting in overoptimization, which tends to add uncalled-for pessimism to the design. Uncertainty numbers should be realistic and not pessimistic for the entire design. Selected timing-critical paths should have greater uncertainty value. Timing exceptions should be reviewed with the corresponding IP owners and IO interface exceptions should be validated.

Every buffer added during the CTS (clock-tree-synthesis) stage for tuning the clock tree should be questioned. Be paranoid about every cell added to the design after CTS and fixing of design-rule violations. Pulling and pushing the clock tree to meet latency and skew targets should be done optimally such that buffers are added in necessary common points only. The clock tree should robust enough to be able to withstand hold-time requirements. Minimal addition of buffers should happen during hold fixing. Ensure the post-clock-tree to post-route correlation is close to perfect. The lesser the jump in delay numbers, the better.

**Checkerboard blockages**

To reduce the cell utilization in a particular region in the sea-of-gates region, you can incorporate hard placement blockages in the form of a checkerboard ( ). As cells won't be placed in regions blocked by the checkerboard, the effective utilization goes down. However, this is not a very favorable solution, as the blockages limit sites available for cell placement. Also, timing may be affected if the paths are timing-critical. Instead, prefer the approach of cell/instance padding.

Sometimes, improper quality of the netlist can lead to increased congestion. As a check, make sure that there are no assign statements in the logical synthesized netlist used for physical synthesis. The
synthesis tool will consider assigns to be direct wire connections, and no optimization will occur on these nets. If some of these nets get covered in timing-critical paths, an unnecessary increase in placement density can occur. Also, scan connectivity in the synthesized netlist should be proper. Congestion-aware scan reordering can resolve congestion due to scan nets.

To finalize on the routing strategy, you still need to deliberate a few important considerations. Needless to say, you should try to achieve the maximum possible timing correlation from the preroute to the postroute stages of the design. For avoiding additional design-rule violations, nets should be routed as straight as possible (minimum distance). Nets that are timing-critical should also follow this approach to negate the net-delay overhead. However, in a congested design, to avoid shorts and other DRC violations, nets try to take a longer path, thereby compromising on timing as well. Timing-critical nets and clock nets should have a higher priority than other nets and should be routed such that the routed wire lengths are smallest. This can be achieved if they are routed first or if the router supports a priority-wise routing; such nets should be given a higher priority. If there is a lot of congestion in a particular spot in the design, the non-timing-critical nets could be rerouted via a longer path; those nets could be selectively detoured and rerouted.

To summarize and conclude, it is imperative to state that tackling congestion is one of the most formidable design challenges, and every designer should aim for resolving such issues at the earliest to avoid delays in the execution cycle. The discussed scenarios capture most of the common congestion problems. The solutions have been tested successfully on designs in sub-90-nm technology nodes and should resolve most congestion bottlenecks.