Congestion can be challenging to understand, due to the complexity of larger designs.

This article covers methodology and tools for overcoming congestion in Vivado designs.

Solution

Finding congestion:

Messaging - Messaging within route_design should indicate when there are high levels of congestion within the design. Below is an example:

```
INFO: [Route 35-449] Initial Estimated Congestion
```

	Global Congestion		Long Congestion		Short Congestion	
Direction	Size	% Tiles	Size	% Tiles	Size	% Tiles
NORTH	8x8	0.49	8x8	1.14	16x16	2.84
SOUTH	8x8	0.52	8x8	1.00	16x16	2.27
EAST	16x16	0.66	8x8	1.07	32x32	5.64
WEST	8x8	0.63	4x4	0.86	16x16	6.20

```
INFO: [Route 35-448] Estimated routing congestion is level 5 (32x32). Congestion levels of 5 and greater can reduce routability and impact timing closure.
```

In this case, the messaging indicates a congestion level of 5 (2^5) or a 32x32 area of tiles that have over 100 percent routing resources used. Higher than Level 4 can generally be a problem.

Metrics - From an open routed design, the Congestion Metrics (right-click within Device view > Metric Vertical/Horizontal routing congestion per CLB) can show you the location of the congestion.

When the metrics view is brought up, the Metric Results will show the congestion percentage (%) and the tile in question. A value over 100 would mean that over 100 percent of the routing resources in this tile are used, and the router will have to divert routes that would normally use this tile around this area.

Diagnosing congestion:

Utilization - One of the main reasons for congestion is the utilization of a design. Designs that have high utilization will be more susceptible to congestion. Generally, a design that is over 80% utilized (Slice LUTs) will become very difficult to route and meet timing. The actual percentage where this difficulty is seen is highly design dependent, and is affected by such factors as the number of control sets (clock, reset, and enable) in the design, and high fanout nets. Also, over-utilization of other site types such as FFs, LUTRAMs, block RAMs, and DSP sites can also lead to congestion.

One suggestion to overcome such congestion would be to balance the utilization of different types of sites. For instance, if many LUTRAMs are causing an over-utilization of LUTs, then moving some of these to block RAM sites could help. Please review the Control Signals and Control Sets section of the Vivado UltraFast Design Methodology Guide (UG949) for guidance suggested utilization and control set levels.

report_high_fanout_nets - Finding high fanout nets can be crucial in fighting congestion. Specifically, non-clock control signals that have a high fanout can cause congestion. You can make a list of high fanout nets with synchronous drivers and use the following command to relieve congested areas.

```
phys_opt_design -force_replication_on_nets
```

If a high fanout net is driven from a LUT and cannot be replicated with phys_opt_design, then this can either be manually replicated in the RTL, or a global buffer (BUFG) can be added if the added delay is acceptable.

report_design_analysis - The report_design_analysis *-complexity* command can also be used to see if a design is more complex and susceptible to congestion.

report_design_analysis -help gives additional information on how to use this. From the IDE, navigate toTools -> Report -> Report Design Analysis.

Example command:

```
report_design_analysis -congestion -complexity -hierarchical_depth 10
```

This will analyze the complexity "Rent" and specify this for 10 hierarchical levels. The -congestion option will also list the most heavily utilized routing tiles.

report_qor_suggestions - Running the report_qor_suggestions command on a routed design can give valuable feedback on constraint, Tcl, and design changes that can help with congestion problems. The focus on the command is QOR improvements relating to timing critical paths, but there are congestion specific suggestions, when specific congestion scenarios are detected.

Logic related to Congestion - Find the logic that occupies the congested tiles by building a schematic. This logic can be checked for connecting high-fanout nets directly related to this congestion. See (Xilinx Answer 66698) for more on this process.

Local vs Global Congestion - Congestion can be local to a certain region of the device, even when the overall device utilization is low. In this case, certain restrictions such as I/O connections or area constraints such as pblocks can cause the congestion and should be checked. Try loosening or removing pblock constraints to see how the results differ.

Check the Clock Placer Floorplan - Check the report_clock_utilization report for clocking floorplans that do not use enough resources for a specific clock. To do this, make a note of all the global clocks entering the clock region with congestion. Are there clocks that use too few clock regions? The placer might not be allocating enough clock regions for a specific clock. To work around this, you can edit the current clock floorplan to add another clock region which could alleviate the congestion. See (Xilinx Answer 66386) for more details on this.

Methods for reducing congestion:

Strategies & directives

- Add phys_opt_design to the implementation flow. This will do timing based physical optimization which can help with congestion. Multiple iterations of phys_opt_design can also help, with each using different options.
  Also, there is the option to use phys_opt_design post-placement or post-routing. See phys_opt_design -help for more information

- Vivado has several congestion specific Strategies that can be used (Tools Options -> Strategies). From these Strategies, specific directives for sub-steps such as place_design & route_design can be found that can be useful for congestion.
  Also, the -Explore directive will generally give better results at a cost of increased run-time.

- Try using the Vivado Synthesis AlternateRoutability directive.

- Try iterating through different place_design directives (found with place_design -help). Specifically, the SpreadLogic_high/medium/low and AltSpreadLogic_high/medium/low directives are meant to spread logic to prevent congestion.
  Comparing the results with different place_design directives and running through route_design will give you an idea of which directives perform better for a specific design.
  Please note that the best performing directive can change as the design changes.
  "report_timing_summary" and "report_design_analysis -congestion" can be used to compare the different directives.

- Finding a suitable opt_design directive is also helpful for a congested design. Run report_utilization after each iteration to see which yields the lowest LUT & FF count, depending on the which element is highly utilized.

- Try using the Vivado Synthesis option -resource_sharing on value. This can often share arithmetic operators and the default is set to auto.

- Try taking certain timing critical paths and over constraining them only during place_design and phys_opt_design. This prioritizes these paths which can lead to better QOR.

- Try floorplanning block RAMs or DSPs. It can be helpful to floorplan block RAMs or DSPs using the best timing results from iterating through different place_design directives, and fixing these LOC constraints for further implementation runs.
  There are a few options to obtain these constraints:

1. From a routed design, select the Block RAMs or DSPs that you wish to fix, and right-click within the Device Window and select "Fix Cells". Saving the design at this point will save the fixed constraints.
2. To print out LOC constraints for the current block RAM or DSP placement, use the below commands. The results can be copied into an XDC file.
   Similar syntax can be used for DSPs, and the search pattern can be found from Vivado IDE Find window.

```
set BRAMS [get_cells -hierarchical -filter { PRIMITIVE_TYPE =~ BLOCKRAM.BRAM.* } ]
```

```
foreach i $BRAMS {puts "set_property LOC [get_property LOC [get_cells $i]] \[get_cells \{${i}\}\]"}
```

Reducing & Controlling fanout

- Use the -force_replication_on_nets option of phys_opt_design. This is a good option for reducing fanout as a post-place phys_opt_design will be able to use the placement information to decide which nets are driven from replicated drivers so that the path length is not excessive.
  Example command: "*phys_opt_design -force_replication_on_nets ${hi_fanout_nets}*" where hi_fanout_nets is a list of nets you would like to reduce the fanout of.
  See phys_opt_design -help for more information. A Tcl example is attached to this Answer Record that finds synchronously driven high fanout nets and creates a variable with a list of nets that can be used in the phys_opt_design command.

- Use the -fanout_opt place_design option. Available with 2017.x, the -fanout_opt performs high fanout replication of critical nets. See place_design -help for more information on the option.

- Use global buffers on non-clock high-fanout nets. The opt_design command can automatically insert BUFGs on high fanout nets.
  Using global clocking resources can help congestion due to high fanout nets. Consult the report_high_fanout report from the routed design to see if there are potential candidates. Also, automatic BUFG insertion by opt_design can be adjusted. See (Xilinx Answer 54177) for more information.

Reducing Local Congestion

- Try reducing/removing LUT combining from synthesis (-no_lc). This can reduce the number of nets entering CLBs that become congested due to LUT inputs.

- Try enlarging or removing pblock constraints if the logic constrained is related to the congested region. This gives the placer more flexibility to avoid the congestion.

Reducing utilization

- Analyze the necessity of resets. Some reset signals might not be necessary if the GSR can provide the necessary initialization. Consult Chapter 4 (RTL Coding Guidelines) of (UG949) for more information on this.