WP381 (v1.0) October 28, 2010

Virtex-6 FPGA Routing Optimization Design Techniques

By: Michelle Fernandez and Peggy Abusaidi

With the ever-increasing need for high bandwidth, system designers continue to increase resource utilization when designing with Virtex®-6 devices. This can sometimes lead to routing challenges and congestion that can impact design closure.

This white paper provides recommendations to help customers mitigate routing challenges in their Virtex-6 FPGA designs.

Xilinx continues to refine the place and route algorithms in each of the ISE® software releases and provides up-to-date information on additional design techniques to help customers optimize routing in their Virtex-6 FPGA designs, making it easier to reach performance and power design goals and achieve next-generation bandwidth requirements.

Introduction

Enhanced CLB, LUT, and Routing Architecture in the Virtex-6 Family

The primary design objectives for the Virtex-6 family were to increase performance and to lower power consumption while ensuring the end product could be produced at a competitive cost. Since its introduction in early 2009, the Virtex-6 family has been well accepted in the industry and has been designed into hundreds of systems across a wide span of applications.

The configurable logic block (CLB) can perform logic, arithmetic, memory, and shift register functions, providing an extremely flexible combination of resources.

Each CLB contains two slices. Slices come in two types: those that are capable of performing logic and memory functions (SLICEM), and those that are capable of performing logic functions only (SLICEL). Each CLB contains either one SLICEL and one SLICEM, or two SLICELs, thus providing enough distributed memory capability at an optimal combination of die size, cost, and power consumption.

Each slice contains four 6-input look-up tables (LUTs) and eight flip-flops. The Virtex-6 family leverages the proven 6-input LUT first introduced in Virtex-5 devices to increase logic capability and reduce levels of logic (and, therefore, reduce delay within a design).

As with the rest of the architecture, local routing in the Virtex-6 FPGAs has been improved over that implemented in Virtex-5 devices, resulting in increased performance and reduced power consumption. There are three types of local routing between CLBs — single, double, and quad — which connect to one, two, and four CLBs away, respectively. Long lines connect across greater distances, providing efficient, flexible connectivity between the resources within the FPGA. Changes in the way the routing is accessed within the device have been validated to improve local routing performance while providing power reduction when compared to an equivalent Virtex-5 design.

The Challenge of Design Congestion

However, even as the Virtex-6 FPGA routing and logic resources become more efficient, customer use cases are becoming more complex, demanding that large amounts of data be buffered inside the FPGA to support the high bandwidths generated by SerDes and parallel I/O architecture. The need for ever-increasing utilization of logic, block RAM, and DSP resources has led to cases where routing challenges occur, with routing congestion or timing errors being the result. Designs with a high number of control nets or non-clock nets with high fanout make the completion of design routing more challenging.

Xilinx recognizes the importance of a routing-optimized design and has a continuous focus on fine tuning software algorithms towards the goal of routing optimization. This white paper is aimed at helping the designer recognize the symptoms of a congested design. Tips are then provided on design techniques and methodologies that can potentially resolve routing challenges, allowing the designer to focus on meeting performance and power requirements.

Symptoms of a Congested Design

The following are indicators that a design might be congested:

1. The design fails to route with no warnings in PAR, detailing specific nets that could not be routed
2. PAR issues the following warning that the design is congested:

   PAR Warning #464

   The router has detected very dense, congested design. It is extremely unlikely the router will be able to finish the design and meet your timing requirements. To prevent excessive run time the router will change strategy. The router will now work to completely route this design but not to improve timing. This behavior will allow you to use the Static Timing Report and FPGA Editor to isolate the paths with timing problems. The cause of this behavior is either overly difficult constraints, or issue with the implementation of synthesis of logic in the critical timing path. If you are willing to accept a long run time, set the option "-xe c" to override the present behavior.
3. Many "intermediate status" stages are reported by PAR:

   intermediate status: 10600 unrouted: Real time: 3 hrs 11 mins 59 secs
   intermediate status: 10719 unrouted: Real time: 3 hrs 41 mins 51 secs
   intermediate status: 10743 unrouted: Real time: 4 hrs 11 mins 47 secs
   intermediate status: 10691 unrouted: Real time: 4 hrs 11 mins 44 secs

Recommendations for Improving Congested Designs

Here are some recommendations that can help when a design is experiencing routing congestion:

1. If area groups, block RAM, or DSP placement constraints exist for the design, remove these from the UCF and rerun.
2. Run SmartXplorer with the congestion reduction strategies (Command Line users: **`-cr`**). To learn how to use SmartXplorer, see UG628, *Command Line Tools User Guide*.

   Note: These strategies typically have a negative impact on timing. If any of the strategies succeed in routing the design, it could indicate that the timing requirements for the design are contributing to the design being congested.
3. Run more than ten cost tables to see how consistent the congestion issue is. If a cost table is found where the congestion is greatly reduced or does not exist, compare that placement in the PlanAhead™ design analysis tool using the Congestion Metric Maps to the placement of a run where congestion does exist. Placement of a certain type of component — block RAM, DSP, distributed RAM, or carry chains, for example — might be contributing to the congestion. Capture the placement of the problem components from the non-congested run and apply it to future implementations.

   Capturing an advantageous placement of components from one run and applying it to a new run, is described in the "Re-use Flow" section of UG633, *Floorplanning Methodology Guide*.
4. Import the placed and routed results in the PlanAhead tool and analyze the vertical and horizontal routing congestion using the Congestion Metric Maps. The PlanAhead tool congestion maps can show "hot spots" in the design related to congestion. To learn how to import the placed and routed results into the PlanAhead design tool, see the "Analyzing Implementation Results" chapter in UG632, *PlanAhead User Guide*. To learn more about accessing the congestion metric maps, see the "Displaying Design Metrics" in the *PlanAhead User Guide*. Here are

some of the key things designers should look for when evaluating the congestion metric maps:

a. Identify whether all of the instances located in the “hot spots” belong to the same hierarchy. If not, consider using area groups to floorplan overlapping hierarchies into separate physical areas, assuming this makes sense from a timing point of view.

b. Evaluate whether or not the slices in question are densely packed, and if so, see if in many cases both outputs of the LUT are being used. This could be an indicator that LUT combining in synthesis/map could be contributing to congestion and ought to be disabled.

c. Evaluate what types of primitives are showing up in the hot spots and what these instances are connecting to by selecting the instances and drawing a schematic of them. In particular, look for distributed RAM or logic connecting to block RAM/DSP. This might be an indication that there are placement issues with the distributed RAM, block RAM, or DSP instances.

   Also look at the nets connecting to the instances in these hot spots. See if they are higher fanout, and evaluate the placement of all of the loads. If the loads are placed far away from each other, consider area grouping them closer together or minimizing the fanout on the net.

d. There is a block of embedded IPs (configuration, system monitor) located in the center of the die floorplan that creates a “hole” in the FPGA logic. Identify whether or not the congestion hot spot is located next to this configuration hole. If it is, try using the environment variable “UAP_DENSMAP_CFG_NEIGHBORHOOD_SLOPE=1”.

e. Evaluate pinout and GT placement in the PlanAhead tool to see if it is causing logic to be spread out on the device. Consider doing a run without pin locations to see if that is contributing to congestion.

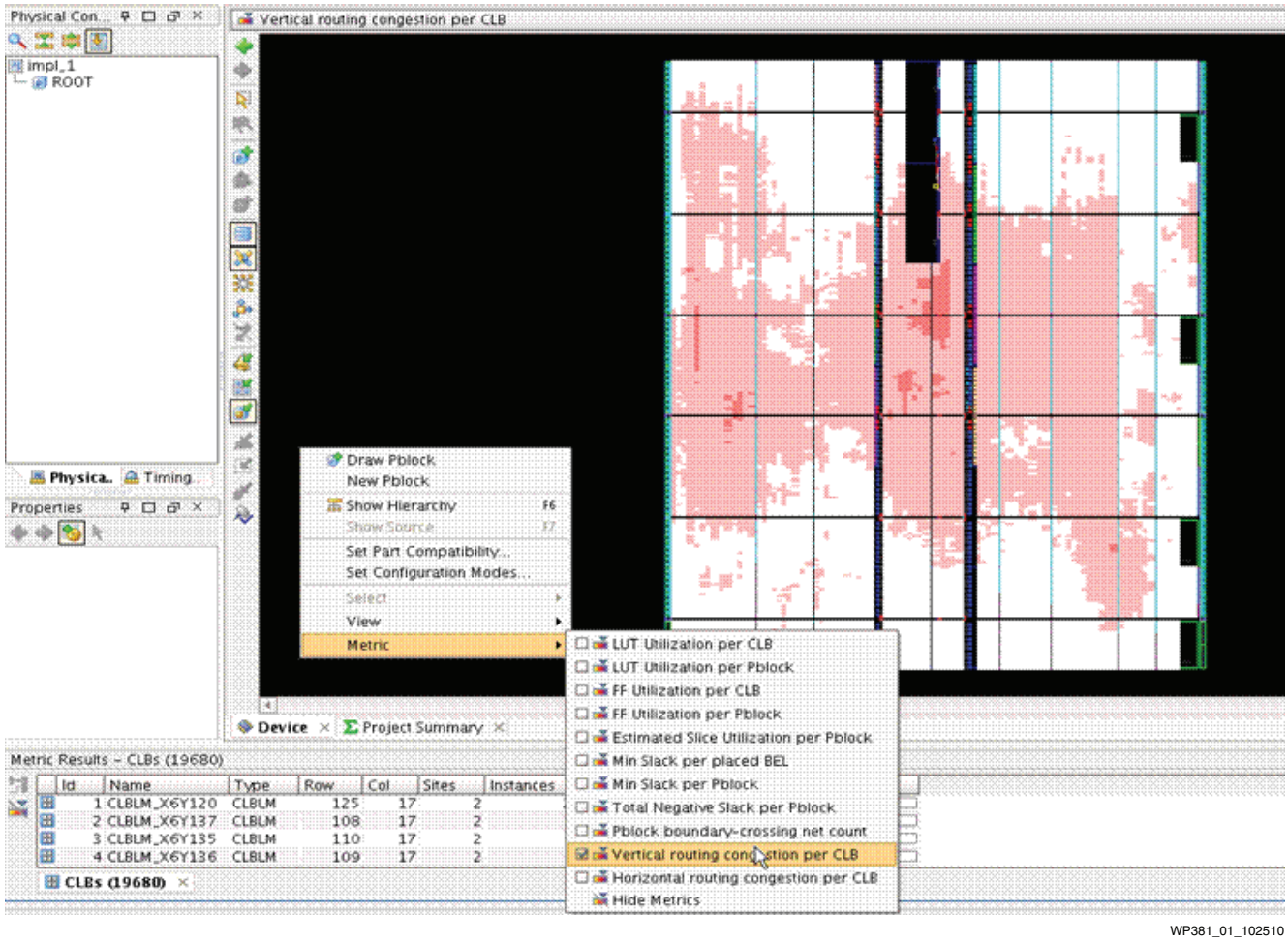


Figure 1: **PlanAhead Tool Interface**

5. Load the design in the PlanAhead tool or FPGA Editor to evaluate the nets in the design by fanout.
   a. For high fanout control signals:

      If there are nets with fanout >1,000, evaluate to see if these are high fanout resets or clock enables. If there are high fanout resets, consider the need for these resets. See WP248, *Retargeting Guidelines for Virtex-5 FPGAs*, which details the Xilinx reset recommendations. If it is a high fanout reset or clock enable that is required, consider placing this net on a BUFG resource. If a high fanout reset has been replicated by synthesis due to fanout control, consider undoing the replication and applying the single reset on the BUFG. Use the following guidelines when applying a high fanout reset/CE on a BUFG resource:

      - A maximum of two non-clock nets in a design can be put on BUFG resources.
      - If the design uses DSP48E1 resources, do not drive the control signals of the DSP48E1s with a BUFG/BUFR.
      - If the design already has a high requirement for BUFGs due to a large number of clocks, evaluate the clock region report in the MAP log file (*map) first to ensure that a single clock region does not have a large requirement for clocks first. If you see many clock regions where the majority of clocks are used, applying a high fanout reset/CE on a BUFG might not be a good option for the design.
      - Evaluate the timing of the synchronous reset or clock enable considered for application on a BUFG. If the timing requirement is stringent such that the delay

through the BUFG is too much for the requirement, the designer might want to consider relaxing the timing on the reset/CR or not using the BUFG at all.

b. For all other high fanout signals:

Use of synthesis controls such as the Global MAX FANOUT limit or the MAX FANOUT attribute in synthesis to replicate the signals should be avoided. Instead, manual replication should be done to ensure that replication occurs at the proper boundaries. In addition, MAP has the ability to do fanout reduction post-placement, and this can be explored. To enable this, the following needs to be done:

- Enable the MAP's register duplication switch:

```
map -register duplication on
```

- In your UCI file, specify the MAX_FANOUT=REDUCE constraint on the net that you need fanout reduction on. The following is a syntax example:

```
NET <net_name> MAX_FANOUT = REDUCE;
```

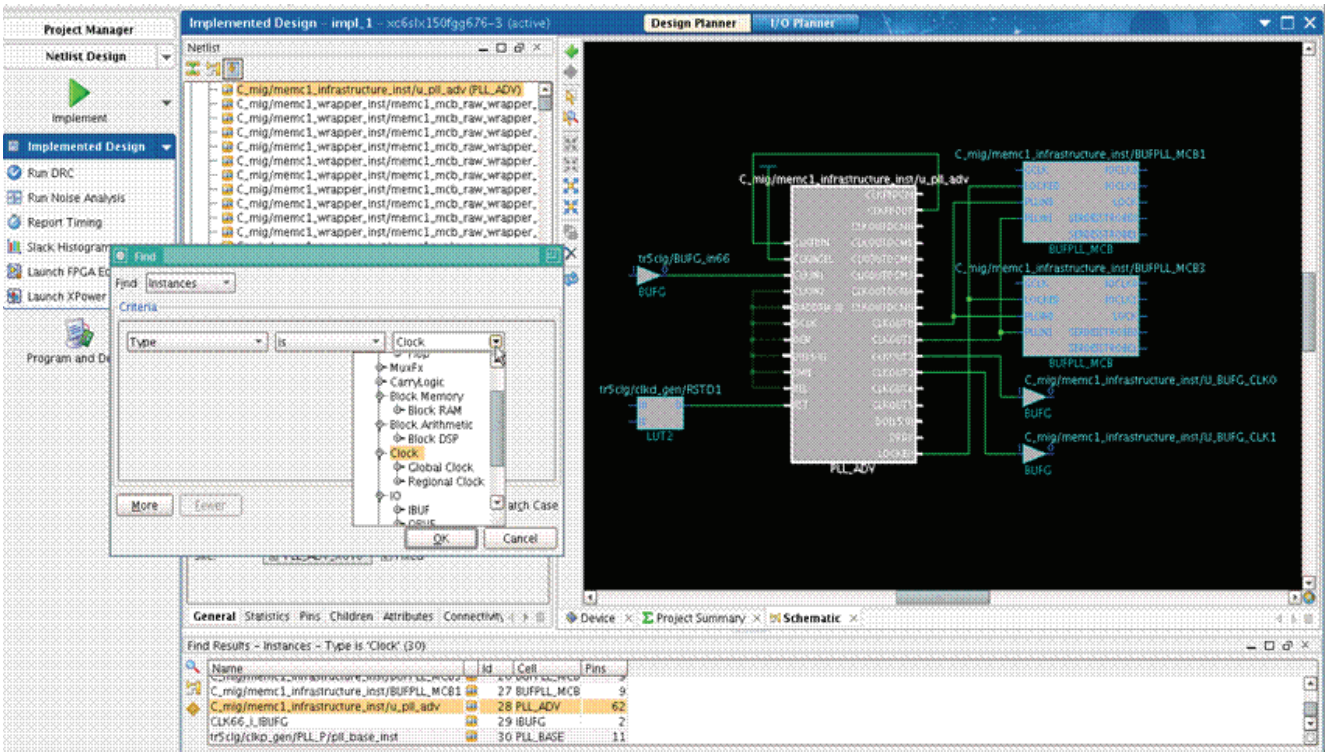
When MAP's register duplication switch is run, a physical synthesis report (`*.psr`) is generated. Review this report to verify that fanout reduction on the net or nets was implemented. If not, manual replication should be implemented.

6. Evaluate block RAM/distributed RAM utilization in the MAP report. If distributed RAM utilization is high (>40%) and block RAM utilization is low (<60%), try moving distributed RAM into block RAM. If block RAM is high, evaluate connectivity and floorplan both block RAM and distributed RAM. Here are some recommendations for floorplanning block RAM/distributed RAM:
   a. Evaluate I/O and data flow in the design using the PlanAhead tool. It also can be useful to import the best placement to see how MAP thinks things should be placed in the design.
   b. Find all groups of four or more block RAMs with shared connectivity.
   c. Using each block RAM's current location, attempt to align them in a pure square where the boundaries are the clock region.
   d. After all groups with over four block RAMs have been laid out, run the tools. The trace report will show groups that cause problems; then the groups can be moved based on the trace report for timing.
   e. The same approach applies for distributed RAM.
7. Experiment with and without LUT combining enabled in synthesis and MAP.
   a. Typically, disabling LUT combining in synthesis and/or in MAP improves routability. You can determine if LUT combining is heavily used in a design by evaluating the MAP utilization report to see how many LUTs are using both the O5 and O6 outputs.

```
Design Summary
--------------
Number of errors:      0
Number of warnings:  436
Slice Logic Utilization:
  Number of Slice Registers:             145,960 out of 470,080   30%
    Number used as Flip Flops:           145,931
    Number used as Latches:                    3
    Number used as Latch-thrus:               26
    Number used as AND/OR logics:              0
  Number of Slice LUTs:                  191,517 out of 239,040   80%
    Number used as logic:                183,551 out of 239,040   76%
      Number using O6 output only:       167,314
      Number using O5 output only:         5,458
      Number using O5 and O6:             10,779
      Number used as ROM:                      0
```

b. Here are the options that control LUT combining in synthesis and MAP:
   - In XST, there is the LUT combining switch (**`-lc auto|area|off`**). By default, it is set to **`auto`**.
   - In Synplify Pro, there is the "Enable Advanced LUT Combining" option. This can be found in the Implementation Options under the Device tab. It is ON by default. If modifying the Synplify Pro project file (*prj), the following is specified **`set_option -enable_prepacking 0|1`** (default is "1")
   - MAP also has a LUT combining option (**`MAP -lc auto|area|off`**). By default, it is OFF. If congestion is experienced while it is set to **`auto`** or **`area`**, disable this option.

8. Other Synthesis Considerations.
   a. When running XST, evaluate the number of wide gates that are getting inferred as carry chains. Use the following switch(es) to control this inferencing.
      - **`-wide_gate_extract (yes|no)`** — Globally disables the mapping of wide gates to carry chains for gates 36 inputs or wider.
      - **`-wide_gate_min_size (default 36)`** — Controls the minimize gate size for when carry chains are inferred
      - **`-wide_gate_max_size (default 500)`** — Controls the maximum gate size for when carry chains get inferred
   b. If running XST, disable "equivalent register removal". By default, it is set to YES.
   c. Both XST and Synplify PRO have "resource sharing" options. By default, they are both ON. Experiment with disabling this option, as it typically improves timing results.
9. Evaluate the clock topology of the design. This can be most easily done in a synthesis schematic viewer or using the PlanAhead tool.
   a. Look to see if there is any redundancy in the clocking structure and if any clocking components can be reduced. By reducing the number of BUFGs/BUFRs in a design, more flexibility is provided to placement.
   b. Look for gated clocks in the design and/or clocks that might be routed on local routing resources.



WP381_02_102510

Figure 2: **Clock Topology**

10. Utilization can also have an impact on congestion. As the device starts to get full, there is less flexibility in how the design can get placed and routed. When utilization is starting to get high, consider the following:
    a. Disable KEEP HIERARCHY options and/or attributes during synthesis to ensure all possible optimizations can be done by synthesis.
    b. Ensure proper inference is occurring, particularly of DSP blocks if utilization of DSP is not already high; if DSP usage is low, determine if more logic can be moved into the DSP blocks. Experiment with XST's “USE_DSP48” option or Synplify PRO's “SYN_DSPSTYLE” attribute to force inference of more logic into DSP blocks.
    c. If LUT Utilization is high, evaluate the number of LUTs used as SRLs. In XST, experiment with the **`-shreg_min_size`** option to increase the minimum shift register size in which SRLs will be inferred, or consider disabled SRL inference altogether with the **`-shreg_extract`** option. Synplify PRO has the **`syn_srlstyle`** option that can also be used to do similar experiments.
    d. See if any asynchronous resets are prohibiting the merging of flip-flops into DSP/block RAM.
    e. For SynplifY PRO users, run without timing constraints to put Synplify PRO in an “Area” mode

Conclusion

With ever-increasing need for high bandwidth, system designers continue to increase resource utilization when designing with Virtex-6 devices. This can sometimes lead to routing challenges and congestion, which impact design closure. This white paper has provided a series of recommendations to help customers mitigate routing challenges in their Virtex-6 FPGA designs. Using a larger device is one way to resolve potential congestion issues as well as leveraging one or several of the design techniques discussed in this white paper. Xilinx continues to develop more congestion-aware algorithms and provides more efficient place and route algorithms in each of the software releases. Xilinx provides up-to-date information on additional design techniques to help customers optimize routing in their Virtex-6 FPGA designs, making it easier to reach performance and power design targets and achieve next-generation bandwidth requirements.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
10/28/10	1.0	Initial Xilinx release.

Notice of Disclaimer