WP311 (v1.2) March 9, 2011

Improving Performance in Spartan-6 FPGA Designs

By: Frédéric Rivoallon

Several considerations need to be taken into account to improve the performance of Spartan®-6 FPGA designs. This white paper discusses how synthesis and implementation can help to optimize design performance.

IP Blocks and Synthesis

Any LogiCORE™ IP, Alliance Core, or other IP cores used in the design are configured for the Spartan-6 architecture. However, blocks intended for a different architecture (e.g., Virtex®-5 FPGAs) can go through the same tools but do not yield the best results.

Synchronous resets should be used in the RTL whenever possible because they are better suited for digital signal processing (DSP) and block RAM inference for Spartan-6 FPGAs. Synchronous resets make it possible for the synthesis tool to pack registers into DSP blocks and block RAM.

The number of logic levels is a good indicator of future performance before moving the design to place and route. Synthesis results should thus be checked to evaluate this number. Higher performance is enabled by fewer levels of logic between registers. Timing constraints should be used during synthesis because they enable more optimizations. To further improve synthesis, additional design approaches can be used, such as register balancing (retiming) or state machine optimizations. Refer to the synthesis tool documentation for more information about these additional design options.

The synthesis tool should be set up so that it reads the black-boxed netlists or IP cores by adding them to the synthesis project (use the read_cores option in XST). This could also improve performance by enabling logic optimizations around the black boxes.

Area optimizations can sometimes lower performance; this is true for the LUT-combining option that groups pairs of LUTs onto the dual-output LUT of Spartan-6 FPGA slices. This option is selected by default in Synplify Pro and should be deselected if high performance is required (see Figure 1).

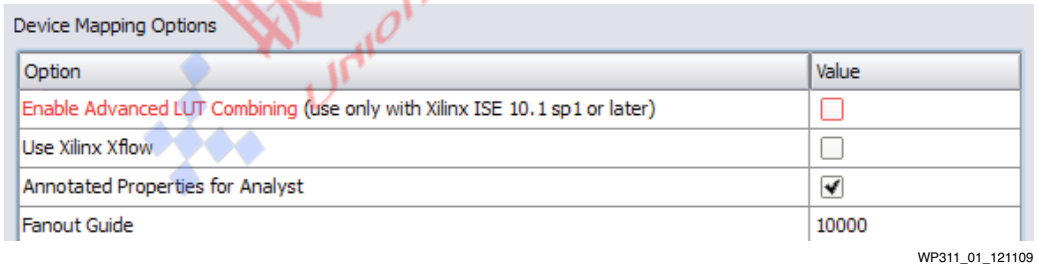

Figure 1: **Implementation Options in Synplify Pro**

For XST, LUT combining can be controlled via the **–lc** option (see Figure 2).

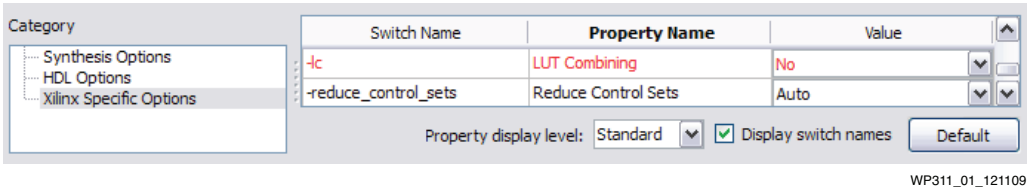

Figure 2: **Process Properties in XST from Project Navigator**

If the critical path spans several logic blocks of the design, the netlist should be flattened so that optimizations can be performed across block boundaries.

Implementation

The timing constraints selected for the design should be realistic. The design should first be run without special options. The results can then be evaluated. If the design does not meet timing but is close to the target timing parameters (with a timing score of 100,000 or less), these options should be implemented:

- Set the LUT combining option of MAP (**–lc**) to **OFF** to ensure that the area optimizations are turned off. Synthesis also performs some LUT combining (as explained in IP Blocks and Synthesis, page 2).
- Set advanced netlist optimizations, such as global_opt, to the speed value. This option re-synthesizes the logic into the netlist and applies fanout optimizations.

Trying these implementation options individually can be time consuming. To make the process faster, the SmartXplorer tool by Xilinx should be used because it allows sets of options to run in parallel. This tool is available from the command line or in the ISE® software Project Navigator environment.

Tightly packed designs should be avoided by keeping LUT utilization below 80%. This helps the place and route tools to find a solution more quickly and avoid performance degradation.

Analysis of Results and Corrective Measures

If the results from place and route indicate that the design has too many levels of logic, the design can be reverted to the RTL to add levels of pipeline or to use retiming. If the results show critical paths dominated by long routing delays:

- Check the path graphically in the design tools (using the PlanAhead™ software) and ensure that place and route is not forced to route around GTP transceivers or the integrated Endpoint block for PCI Express® technology (see Figure 3).
- If the net with the long routing delay has a high fanout, evaluate whether this fanout can be reduced. Replicating the source of the net can help with this fanout reduction. The design can also be floorplanned to gain control over the placement of logic.

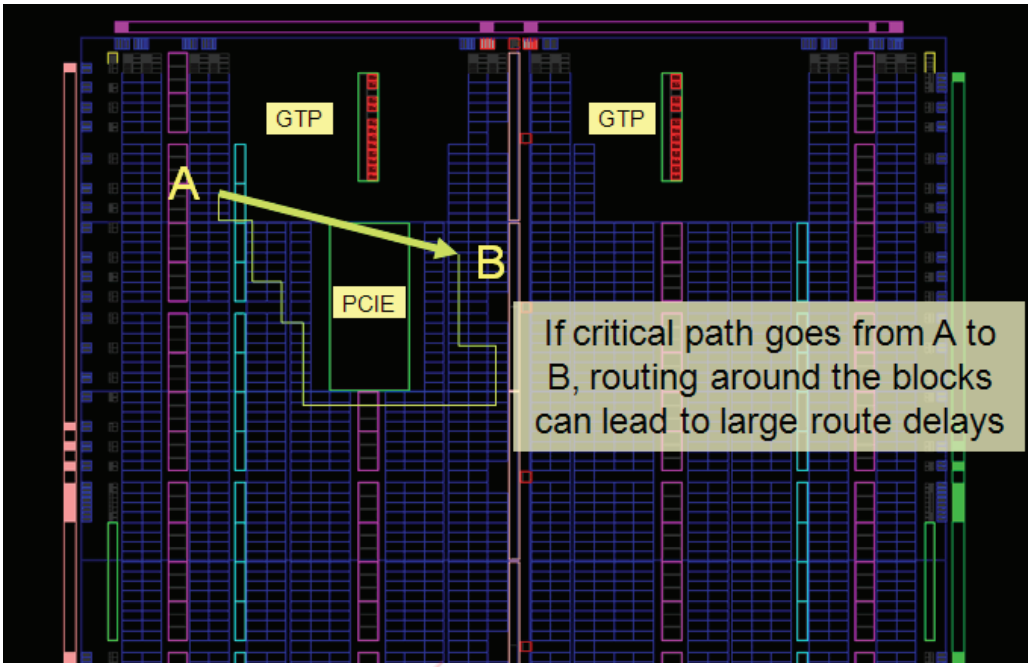

Figure 3: **Floorplan View (PlanAhead Software)**

If the critical path is related to an I/O, the I/O register should be used. If this register is already being used, the I/O should be placed closer to its target. To have better access to the FPGA logic, critical I/Os should not be placed next to dedicated IP blocks.

If the critical paths are related to blocks such as the DSP48 or block RAM, these paths should be fully pipelined. If they are already pipelined but still show in the design critical paths, they can be floorplanned or locked down (with LOC constraints) to ensure that they are placed optimally.

Critical paths in the design should not require the routing to go around the integrated Endpoint block for PCI Express technology or the GTP transceiver blocks in the center top and bottom halves; critical routes that must travel over these blocks incur a significant delay penalty (see Figure 3).

Summary of Tips and Guidelines

This section summarizes the design tips discussed in this white paper:

- Planning and design setup:
  - Plan I/Os carefully using the pin placement and DRC tools in the PlanAhead software.
  - Use IP optimized for Spartan-6 FPGAs.
  - Avoid asynchronous resets in the RTL.
  - Use a -3 speed grade device for better performance. (This gives 30% faster performance than a -2 speed grade device.)
  - When floorplanning, ensure that the area groups allocated for the critical logic are large enough to ensure more logic packing and routing possibilities for the software.

- Optimization:
  - Apply timing constraints in synthesis and use advanced options such as retiming to improve timing.
  - Turn off LUT combining. (This gives the software more options to find faster connections and route the design.)
  - Use SmartXplorer to run special options.
  - Limit large fanout nets (if practical).

Conclusion

The performance of Spartan-6 FPGA designs can be improved by leveraging performance options in the design tools (ISE software and SmartXplorer) and understanding the bottleneck(s) in the design. Using a faster speed grade device and optimizing the RTL are other simple and proven methods of improving design performance. For more information about targeting (or retargeting) designs to Spartan-6 FPGAs, refer to WP309, *Targeting and Retargeting Guide for Spartan-6 FPGAs*.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
03/01/10	1.0	Initial Xilinx release.
05/25/10	1.1	Removed DSP48A1 from IP Blocks and Synthesis. Updated -2 speed grade in Implementation. Updated Conclusion.
03/09/11	1.2	Updated the Implementation section.

Notice of Disclaimer