Guided Design Using BLIS

With Block Level Incremental Synthesis (BLIS), your design implementation times will improve dramatically.

by Karen Fidelak
Technical Marketing Engineer, Xilinx
karen.fidelak@xilinx.com

Incremental design changes (due to ECOs, specification changes, and repeated design iterations) can cause significant delays if you have to synthesize and place and route your entire design after each change. Ideally, your synthesis and place-and-route software tools should recognize where changes have been made in your overall design and recompile just those portions that have changed. That's what you get with BLIS, a unique synthesis and place-and-route capability, developed by Synopsys for Xilinx, that provides a guided synthesis methodology. Used in conjunction with Xilinx High-Level Floorplanning, BLIS provides the most robust incremental design capability ever offered.

BLIS, a part of the Synopsys FPGA Express/FPGA Compiler II v3.4 software (FE/FCII), is now available in the Xilinx ISE 3.2i development tools.

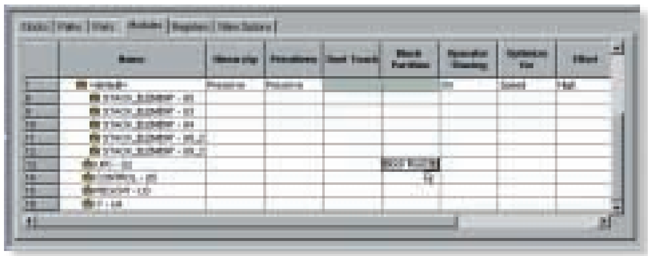

Figure 1 - Constraint Editor, specifying Block Roots

Block Level Incremental Synthesis

As you make design changes, BLIS recognizes "blocks" of the design which have been changed at the source, and intelligently synthesizes only those portions of the design. In this flow, a block is defined as a module/entity and any hierarchy tree beneath it. To enable BLIS, you choose blocks in your design that you want to denote as "Block Roots" through the FE/FCII Constraint Editor GUI or scripting language, as shown in Figure 1.

A Block Root is a block which is intelligently updated by FE/FCII in incremental synthesis runs, and has the following characteristics:

- A separate netlist is created by FE/FCII for each Block Root.
- Only those Block Roots whose corresponding source has been modified are re-synthesized.
- The Block Root has hard boundaries around it–no optimization occurs with neighboring modules.

The Advantages of BLIS

There are two main advantages to using this type of incremental flow.

- Runtime for both synthesis and place-and-route will be improved because only the modified portion of your design will be re-synthesized and re-netlisted. The remainder of the design will remain unchanged and the netlists for the unchanged portions of the design will not be rewritten. Because the netlists of the unchanged portions of the design remain untouched, you are assured that all net and instance names in that part of your design are identical to earlier runs.
- Timing predictability will be improved because the "Guide" function of the place-and-route tools, which relies on matched component names from run to run, will have a higher success rate.

Benchmarks

We compared the results of incremental design flows using BLIS against the more traditional methodology of re-synthesizing and re-routing the entire design. With the BLIS flow, incremental changes are made to a small number of design blocks (Block Roots). With the traditional flow incremental changes are made to the same design blocks, however they are not specified as Block Roots.

After our example design synthesis was completed, the design was placed and routed using the Guide feature of the Xilinx implementation tools, which allow you to specify an existing placed-and-routed design to be used as a "Guide" when implementing a design. The existing placed-and-routed design was used as a template when re-implementing the design. Any portions of the design which existed in both the "Guide" design and the new modified design (determined by matching net and component names) were placed in the same location in the new implementation as they were in the "Guide" design. New or changed logic was implemented around existing, "Guided" logic.

Runtime Improvements

Runtime improvements of up to 50% (with an average of 47%) were observed when using BLIS with Xilinx Guided Place-and-Route in an incremental design flow;

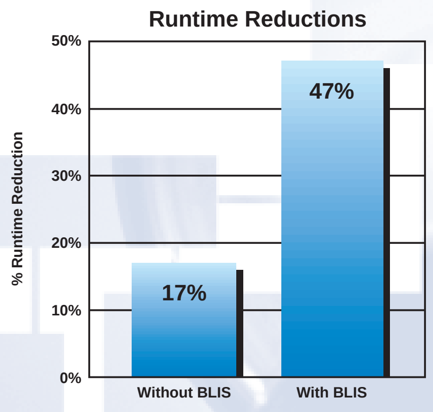

Figure 2 - BLIS runtime reductions

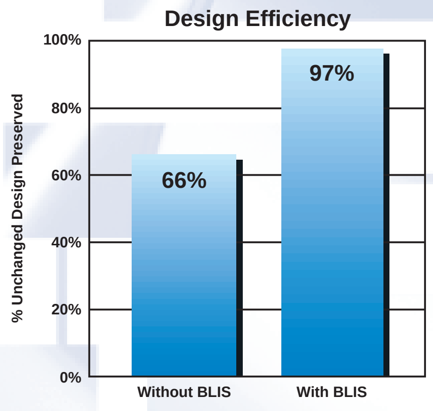

Figure 3 - BLIS design efficiency

Figure 2 shows averaged design results. Because FE/FCII does not re-elaborate or re-optimize unchanged blocks of the design, synthesis runtime was reduced.

Implementation runtime was improved due to increased design component matching during guided placement and increased signal matching during routing. Additionally, the synthesis tool does not rewrite the EDIF netlists for the unchanged blocks, further reducing runtime, because no file re-translation is needed.

Guide Improvements

When a design is placed-and-routed using the Guide feature, the success of the Guide can be determined by the "Design Components Matched" statistics available in the Place-and-Route report. The higher the percentage of matched components, the closer the incremental design is to the original results, leading to better predictability of timing and placement results.

When using the BLIS incremental design flow, Guide success rates reached levels of at least 95%, and averaged 97%. When BLIS was not used to guide the design, component and route matching was as low as 52%, as shown in Figure 3.

The improvements when using BLIS can be attributed to the increase in net and component name matches between the original placed-and-routed design and the incrementally modified version of the design. Because unchanged blocks of the design are not re-synthesized, the netlists are untouched and thus remain identical to the original version. (Even if there are no logic changes in the source, re-synthesizing a block can lead to net and component names being changed in the final netlist.)

Conclusion

When utilizing FE/FCII Block Level Incremental Synthesis in a Xilinx guided design, runtimes as well as timing and placement consistency exhibit significant improvements over a more traditional design flow. These enhancements help you achieve a higher level of productivity by allowing you to synthesize and implement incremental design changes, with a significantly reduced runtime, while preserving the unchanged portions of your design. This new design flexibility allows you to realize the productivity necessary to complete large or small FPGA designs faster.