Using Block-Level Incremental Synthesis in FPGA Compiler II and FPGA Express

Block-Level Incremental Synthesis allows you to modify a subset of a design and then re-synthesize just the modified subset.

by Alan Ma
Senior Corporate Applications Engineer, Synopsys, Inc.
alanma@synopsys.com

Advancements in the capacity and speed of FPGAs have made this technology increasingly attractive for million-gate designs. As the complexity of the designs grows so does the need for advanced synthesis and place-and-route tools. One of the highly sought features is the ability to recompile only the modified portion of a design after modifications have been made. Such a feature is needed not only to reduce compilation time, but also to preserve the timing behavior of certain sections of a design when other sections of the design are changed.

To address these requirements, FPGA Compiler II and FPGA Express (FCII/FE) version 3.4 introduce Block-Level Incremental Synthesis (BLIS) to allow you to modify a subset of your design and then re-synthesize just the modified subset. A design can be divided into "blocks," the smallest subset to which BLIS can be applied. FCII/FE generates an optimized netlist for each block, which does not change unless the design associated with that block has been modified. The netlist of each block is then presented individually to the place-and-route tool which is capable of recompiling only the modified netlists. Not only does this increase the likelihood of preserving post place-and-route timing behavior for the unmodified blocks, overall compilation time for both synthesis and place-and-route is significantly reduced.

Identifying Blocks

A block is the smallest subset to which BLIS can be applied. It can be a Verilog module, a VHDL entity, an EDIF netlist, or a combination of these, as long as they form a tree within the design hierarchy. The top-level module/entity/netlist of this tree is, by definition, the block root. The components of a block include the block root and all modules/entities/netlists in its tree of the design hierarchy that do not include another block root.

For example, the top-level design "TOP" in Figure 1 has two subdesigns A and B. A instantiates C and D. B instantiates E and F. By definition the top-level design TOP is a block root. If you also decide to designate A and E as block roots, then the entire design will have three blocks:

- Block 1: TOP, B, F
- Block 2: A, C, D
- Block 3: E

Any modifications to any module/entity/netlist contained in a block cause the entire block to be re-synthesized. For example, if F is modified then every member of Block 1 (TOP, B, F) will be re-synthesized even though TOP and B have not been changed.

Blocks can be identified in both the Graphical User Interface (GUI) and the shell of FCII/FE. In the GUI, right-click an elaborated implementation in the Chips window and select Edit Constraints. Select the Modules

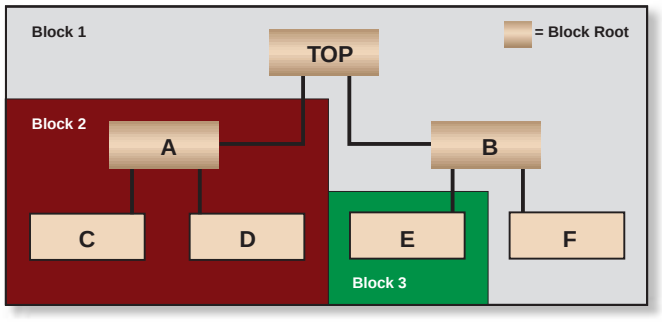

Figure 1 - Example of blocks and block roots

tab to display the Modules Constraint Table. You can then specify any subdesigns as block roots in the Block Partition column. To remove a block root designation, click the particular cell and select Remove. Note that the top-level design is always identified as the

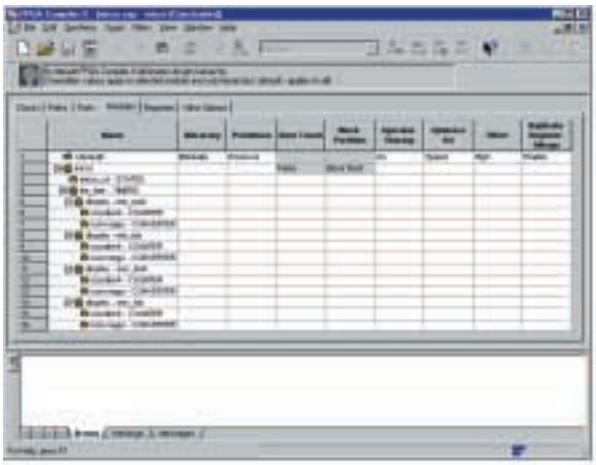

Figure 2 - Modules Constraint Table

block root by definition and it cannot be removed. Figure 2 shows the Modules Constraint Table.

In the shell, use the command set_module_block followed by the option "true" and the path to the module/entity/netlist to be specified as the block root. For example:

```
fc2_shell/fe_shell> set_module_block true /TOP/A
```

In the interactive mode, you can remove any block root designations by using the "false" option. For example:

```
fc2_shell/fe_shell> set_module_block false /TOP/A
```

In the batch mode, the same can be achieved simply by removing the set_module_block commands associated with the block roots to be removed in the fc2_shell or fe_shell scripts.

Please refer to the man page of set_module_block for additional usage and syntax information. To access the man page:

```
fc2_shell/fe_shell> man set_module_block
```

Block roots can only be designated on user-created modules/entities/netlists. For example, attempting to modify the setting on a primitive or the top-level design will result in the following error message:

```
Error: Cannot set block option on module 'AND'
```

Furthermore, the concept of block and block root only applies when the target architecture supports BLIS. Attempting to apply this feature on an architecture which is not supported by BLIS will result in the following error message:

```
Error: block assignments are not supported for the target technology of this chip
```

BLIS is currently available for the Xilinx Virtex architecture.

Design Planning

The advancement in synthesis and place-and-route technologies certainly plays an important role in Quality of Results (QoR) but thorough design planning can never be replaced. In order to take full advantage of BLIS it is important to understand that:

- FCII/FE uses a time stamp to determine if an implementation is out-of-date. If the time stamp of any of the analyzed design source files is newer than the elaborated implementation then the elaborated implementation needs to be updated.
- A block is the smallest subset to which BLIS can be applied and any changes in any member of a block cause the entire block to be re-synthesized.
- A block represents "hard" boundaries which FCII/FE does not optimize across.

Therefore it is recommended that each module/entity/netlist in a design be described in its own design source file. Doing so ensures that modifying a module/entity/netlist will not affect the time stamp of other modules/entities/netlists which could potentially be members of other blocks.

In Figure 3, A and B are described in the same design source file. Modifying A not only causes the entire Block 2 to be re-synthesized, it also affects the time stamp of B for being in the same design source file as A. The newer time stamp of B causes the entire Block 1 to be re-synthesized. Needless to say, this is not a desired behavior.

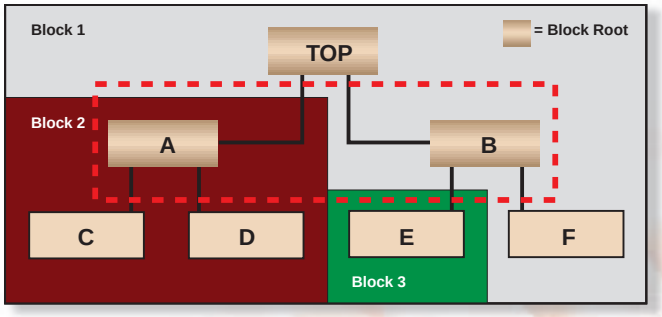

Figure 3 - Example of design planning

It is obvious that BLIS is not useful in the extreme case where all modules/entities/netlists are described in a single design source file.

Since FCII/FE does not incrementally optimize across block boundaries it is important not to break combinatorial logic into different blocks. For best QoR, it is recommended to observe conventional Register Transfer Level (RTL) coding style when partitioning designs. This involves grouping related combinatorial logic within a module/entity and registering all outputs of such a module/entity.

Note that BLIS is most effective when modifications are done within a module/entity. Changing the partition or the pins of the modules/entities of the design causes the entire design to be re-synthesized. This is why thorough planning in the early stage of the design process is vital to success.

Implementation Update

This section describes what needs to be done in the GUI and the shell to update an implementation after a design has been modified. It is assumed that blocks have been defined as described previously, an optimized implementation has been created, and place-and-route netlists have also been generated. For those who are not familiar with the synthesis process of FCII/FE, please refer to the *FPGA Compiler II/FPGA Express Getting Started* manual for step-by-step instructions.

In the GUI, a red question mark next to a design source file in the Design Sources

window indicates that the file has been modified. Right-clicking the modified file and selecting Update File reanalyzes the file for syntax errors.

Similarly, a red question mark next to an elaborated implementation in the Chips window indicates that the implementation is out-of-date with respect to its design source files. Right-clicking the elaborated implementation and selecting Update Chip re-elaborates the implementation.

Once the design has been re-elaborated, right-clicking the optimized implementation and selecting Update Chip re-optimizes the design. If blocks are properly defined as described previously, BLIS will be automatically enabled. In this situation only the blocks that are associated with modified files are re-optimized. Note that there has to be at least two blocks defined to enable BLIS.

Instead of individually updating the design source files, the elaborated implementations, and the optimized implementations, you can update the entire project in one step by right-clicking the project icon in the Design Sources window and selecting Update Project. It is important to understand that FCII/FE generates one netlist in EDIF format for each block defined. The names of the netlists are the same as the module/entity/netlist name of the respective block roots. Performing an Update Project ensures that only the netlists associated with modified blocks are regenerated. Note that explicitly choosing Export Netlist forces all netlists to be regenerated. This step is therefore only recommended when the design is synthesized for the first time.

If Export Timing Specifications is selected in the Export Netlist dialog box, FCII/FE generates a Synopsys Constraint File (.scf) to pass timing constraints entered in FCII/FE to the place-and-route tools. Since the Xilinx place-and-route tool does not read .scf directly, the information in this file must be transferred to a User Constraint File (.ucf) for timing constraints to be considered during place-and-route.

Figure 4 shows a sequence of equivalent fc2_shell/fe_shell commands to be used in the interactive mode. Please refer to the man pages for complete usage and syntax information.

Limitations

As mentioned previously, time-stamping is used to determine if an implementation is out-of-date. FCII/FE processes time stamps in whole seconds. If both analysis and elaboration finish within one second, then the analyzed design source files and the elaborated implementation will have the same time stamp. When the time stamp of the elaborated implementation is older than or equal to the analyzed design source files, the existing elaborated implementation is discarded and re-elaborated upon Update Chip or Update Project. The new elaborated implementation will then have a newer time stamp than the existing optimized implementation. This causes the existing optimized implementation to be discarded and re-optimized when actually none of the design source files have been modified. Because designs in general take longer than one second to be analyzed and elaborated, this limitation should not present any problems.

Note that BLIS is driven only by the difference in time stamps, the existence of at least two defined blocks, and any change of block roots. Other operations, such as modifying timing constraints, do not cause block-level incremental re-synthesis of the implementation.

Conclusion

FPGA Compiler II and FPGA Express version 3.4 introduce Block-Level Incremental Synthesis allowing you to modify a subset of a design and then re-synthesize just the modified subset. They generate an optimized netlist for each block and the netlist of a block does not change unless the design associated with that block has been modified.

The place-and-route tools that support block-level incremental place-and-route are able to recognize and recompile only the netlists that have been changed. Not only does this increase the likelihood of preserving post place-and-route timing behavior for the unmodified blocks, overall compilation time for both synthesis and place-and-route can be significantly reduced.

Please visit http://www.synopsys.com/products/fpga/ for the latest in FPGA synthesis technology.

```
create_project TOP

add_file -library WORK -format VHDL c:/designs/top/TOP.vhd
add_file -library WORK -format VHDL c:/designs/top/A.vhd
add_file -library WORK -format VHDL c:/designs/top/B.vhd
add_file -library WORK -format VHDL c:/designs/top/C.vhd
add_file -library WORK -format VHDL c:/designs/top/D.vhd
add_file -library WORK -format VHDL c:/designs/top/E.vhd
add_file -library WORK -format VHDL c:/designs/top/F.vhd

analyze_file

create_chip -name TOP -target VIRTEX -device V800FG680 -speed -6 -frequency 50 TOP

current_chip TOP

set_module_block true /TOP/A
set_module_block true /TOP/B/E

optimize_chip -name TOP-Optimized

export_chip -dir .

# design has been modified

update_project

list_message
```

Figure 4 - fc2_shell/fe_shell commands