Questions & Answers

From the Xilinx Applications Engineering Staff

by Kamal Koraiem, Product Applications Manager, Xilinx, kamalk@xilinx.com

Virtex

Q: What's the recommended way to asynchronously set or reset flip-flops in a Virtex design, and why? Is it still necessary to use the STARTUP_VIRTEX block?

A: Write this high fan out reset/set signal explicitly in the HDL codes, and do not use the STARTUP_VIRTEX block. There are two advantages:

- This method gives you a faster design. The reset/set signal will be routed onto the secondary long lines in the device, which are global lines with minimal skews and high speed. Therefore, the reset/set signal on the secondary lines is much faster than the the GSR net of the STARTUP_VIRTEX block. Because Virtex is rich in routing, placing and routing this signal on the global lines is easily done by our software.
- Our trace program will analyze the delays of this explicitly written reset/set signal. You can read the .twr file (report file of the trace program) and find out exactly how fast it is. Trace does not analyze the delays on the GSR net of the STARTUP_VIRTEX block. It is not necessary to use the STARTUP_VIRTEX block if the reset/set signal is explicitly coded. However, you still have the option of using the STARUP_VIRTEX block if you want, if GSR speed is not a concern.

Core Generator

Q: I just tried generating a CORE Generator module using the new 2.1i software release. I specified that I wanted a VHDL (or Verilog) behavioral model, but did not get one in my project directory. All I see is a .VHO (or .VEO) file.

A: The HDL behavioral simulation flow is new in the 2.1i version of the CORE Generator. The 2.1i CORE Generator does not generate a .VHD or .V file in the project directory. Instead, it creates a .VHO (for VHDL) or .VEO (for Verilog) template file containing the code snippets required to integrate the core into a higher level design block's behavioral simulation netlist.

In addition, before any behavioral simulation of a core can be done, you must:

- Run the **get_models** utility to extract the models into a separate source library.
- Analyze the library, if required by your simulator.
- Set your simulator to point to the extracted (and analyzed) library.

Refer to the Design Flows chapter of the CORE Generator User Guide (available from within the CORE Generator under Help-> Online Documentation) for more details.

(Continued)

Implementation

Q: I hear that Xilinx has a new command line program that will allow me to run the full suite of Xilinx implementation tools. What is it and where can I find it?

A: This new command line tool is called XFLOW. It allows you to run the full suite of Xilinx implementation, simulation, and configuration flows. It is device independent and has a customizable interface to the Xilinx tools. XFLOW accepts design, flow, and option files as input. This tool is making its debut in the 2.1i release. To run it, simply type in "xflow" from the command line. For detailed information and examples, please refer to the XFLOW chapter of the 2.1i Development System Reference Guide.

Q: Do the implementation tools have a feature that allows me to automatically specify the copying of files into version and revision directories for the purpose of maintaining source netlist information (.EDN, .EDF, .VHD, .VER, .SCH, etc...) along with implementation output?

A: Yes. The Xilinx Design Manager has a "backdoor" feature that allows you to automatically specify the copying of files into each and every newly created version or revision directory. To access this "File List" feature, you must select your project name in the Design Manger window, then right-click and select "Properties". For more detailed instructions, please reference:
http://support.xilinx.com/techdocs/6704.htm

Timing Analysis

Q: What is the Hierarchical Report in the Timing Analyzer?

A: The Timing Analyzer displays FPGA and CPLD analysis reports in a hierarchical format. The report window (used to display all .twr files) now has three panes (Figure1). The lower-right pane is the text view. It just displays the text of the .twr file. The left pane is the index or outline view. Click on the labels to scroll the corresponding line in the report to the top of the text view. Click on the "+" or "-" buttons to expand or collapse topics. The other standard mouse and keyboard manipulations for tree views also work.

The upper-right pane is the context or path view. It shows the path through the topic hierarchy to the item currently selected in the index view. The contents of the context view change only when the selection in the index view changes. Click on items in the context view to scroll the text view to the appropriate place.

Report Window

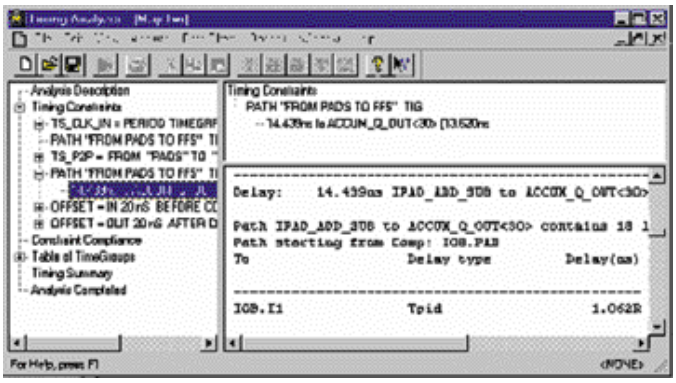

Figure 1

This can be handy if you're deep in a list of paths for a timing constraint and need to jump back to the timing constraint definition.

Q: How do I add a probe in FPGA Editor?

A: The following steps walk you through the procedure to create a probe in FPGA Editor. Also refer to Solution 6616, at: http://www.xilinx.com/techdocs/6616.htm:

1. Launch FPGA Editor either via command line **(fpga_editor)** or the GUI.
2. Load the design's ncd file, either from map (map.ncd) or par (design.ncd), and select the Edit Mode to be "Read Write".
3. Select the "Probes" button on the leftside.

(Continued)

4. Select the "Add" button, to add a probe.
5. Hilight the net that you want to probe, in the "Select Net" box.
6. Choose the "Method," either Automatic or Manual, in the "Select Pin Numbers" box.
   - Automatic - FPGA Editor picks the closest pin to the net to be probed.
   - Manual - You choose which pin is used for the probe.
7. Select the "OK" button, to add the probe.
8. The probe is listed in the "Probes" window with the delay and the pin number.

Q: How do I apply a PERIOD timing specifications in 2.1i when using a Virtex CLKDLL?

A: The rules regarding property tracing through the DLL have changed in 2.1i (Figure 2). When a **TNM_NET** property is traced into the CLKIN pin of a Virtex CLKDLL component, the TNM group and its usage will be examined. The TNM will be pushed through the CLKDLL only if the following conditions are met:

- The TNM group name is used in exactly one PERIOD specification.
- The TNM group name is not used in any FROM-TO or OFFSET specifications.
- The TNM group name is not referenced in any user group definition.

If any of the above conditions are not met, the TNM will not be pushed through the CLKDLL, and the following error message will be issued: ERROR:NgdHelpers:702 - The TNM "PAD_CLK" drives the CLKIN pin of CLKDLL "$I1".

This TNM cannot be traced through the CLKDLL because it is not used in exactly one PERIOD specification. This TNM is used in the following user groups and/or specifications:

```
TS_PAD_CLK=PERIOD PAD_CLK 20000.000000 pS HIGH 50.000000%
TS_01=FROM PAD_CLK TO PADS 20000.000000 pS
```

If the above conditions are met each clock output pin on the CLKDLL will be examined to see if it is connected to a net with at least one other connection (it is not a dangling net). If the output pin has a net, a new TNM group will be created on that net, and a new PERIOD specification will be created for that group. The new specification will be copied from the original PERIOD specification, and then modified as follows:

Output Pin	Modifications to PERIOD Specification
CLK0 CLK90 CLK180 CLK270	If the DUTY_CYCLE_CORRECTION=TRUE property is found on or above the CLKDLL, the duty cycle will be adjusted to 50%. If DUTY_CYCLE_CORRECTION=FALSE, the duty cycle will be unchanged from the original PERIOD specification. If no DUTY_CYCLE_CORRECTION property is found, the default value of TRUE will be assumed.
CLK2X	The PERIOD value will be doubled (if originally expressed as a frequency) or divided in half (if originally expressed as a delay). The duty cycle will also be adjusted to 50%.
CLKDV	The PERIOD value will be divided (if a frequency) or multiplied (if a delay) by the value in the CLKDV_DIVIDE property. If no such property is found on or above the CLKDLL, the default value of 2.0 will be used. The duty cycle will also be adjusted to 50%

Figure 2

(Continued)

- If the original TNM_NET property is pushed only into the CLKDLL CLKIN pin (it does not trace to any appropriate elements without going through the CLKDLL), the original TNM group and the original PERIOD specification will be eliminated from the design.
- If a newly-created TNM group is pushed forward from a CLKDLL output and encounters the CLKIN input of a second CLKDLL (such as in the 4X configuration), the above process will be repeated to fur ther adjust the PERIOD specifications per the behavior of the second CLKDLL.
- If the group created for the first CLKDLL traces only into the second CLKDLL, that group and its PERIOD specification become unnecessary and will be eliminated. For further information regarding property tracing through the CLKDLL refer to the Developmental System Reference Guide Chapter 6.

Board-Level Timing Analysis

Q: Can I do a board-level timing analysis that includes my FPGA as well as other chips?

A: Yes. You can do a board-level static timing analysis using either the Mentor Tau tool or the Viewlogic Blast tool. Both can take in a STAMP file created by the Xilinx tool trace. Please see the following document for trace options and the method to create the stamp files: http://support.xilinx.com/pub/documentation/interfaces/tau_blast.pdf

Virtex Configuration

Q: Is Readback supported in Virtex?

A: Readback in Virtex is supported through the SelectMAP and JTAG ports. Readback may be used for in-system configuration verification, or internal state capturing and logic debugging. Xilinx Application Note XAPP 138 Virtex Configuration and Readback describes the process and procedures for performing readback operations in detail. XAPP 138 also describes the readback files produced by the 2.1i software that are used in readback operations. While the options to produce these readback files are also included in the 1.5i software, some of the resulting files needed for verification are generated incorrectly. Therefore, the 2.1i software is required to support this functionality.

Q: Which BitGen options should be used when configuring Virtex devices in a daisy-chain?

A: When configuring multiple Virtex devices in a serial daisy-chain the DONE pins must be connected together and the DONE_cycle must be first in the startup sequence. This is most easily accomplished by using default settings. If not specified by the user, the settings for DONE_cycle, GTS_cycle, GSR_cycle, and GWE_cycle will default to the startup sequence states 4, 5, 6, 6, respectively.

When the first device has received all of its configuration data it will also have received the command to run through the startup sequence. With all the DONE pins tied together, unconfigured devices will externally hold the DONE pins of configured devices LOW. When the startup sequence releases the DONE pin (DONE_cycle) the sequencer will not progress passed this state until the DONE pin has externally gone HIGH.

If it were allowed to do so, once the GTS_cycle state was achieved, the DOUT pin would no longer be capable of passing configuration data to the daisy-chained FPGAs. This would also be the case if the GTS_cycle was set to an earlier state than the DONE_cycle.