

Digital Duct Tape with FPGA Editor

Xilinx Senior FAE Clayton Cameron shows us his tips and tricks for using his favorite tool in the ISE tool suite, FPGA Editor.



by Clayton Cameron
Senior field applications engineer
Xilinx, Inc.
clayton.cameron@xilinx.com

There comes a time in most design cycles where a little creativity—you might call it digital duct tape—is required to make your design work. Over the past eight years, I've seen some of the best engineers do truly amazing things with this approach, often using one essential tool: FPGA Editor.

FPGA Editor allows you to see your implemented design and review it to determine if that is truly what you wanted at the FPGA fabric level—a must for any engineer or FAE. Let's say you're given a co-worker's design on which to make changes, and their HDL source is hard to understand or there are no source comments or documentation. Maybe you just want to lock down some clock logic but you don't know the instance name or how to lock it in place. Certain tips and tricks for exploring the FPGA fabric and creating command line patches can help you meet fast-approaching deadlines.

General Fabric Exploration

One of the first things I normally do when Xilinx releases the tools for a new FPGA is open FPGA Editor and look at the FPGA fabric. You get there by going to the **Xilinx** → **ISE** → **Accessories** menu and clicking on the **FPGA Editor** icon or by typing `fpga_editor` at the command prompt. After the GUI is open, select **New** under



the **File** menu. FPGA Editor will ask you for a design file name and a physical constraint file. At this point you have no design files, so enter anything for the design file name (for example, `test.ncd`) and select a part type you wish to review. FPGA Editor will use the same file name for the physical constraint file and load a blank design.

Another option is to compile one of the provided ISE® tool suite example designs and load it into FPGA Editor for fabric review. Loading an example design will give you more details and make it easier to locate items of interest.

To navigate FPGA Editor, you really only need to know two things:

1. How to zoom in and out using the **CTRL / Shift** key shortcuts.
2. How to zoom to selected items using the **F11** key.

To zoom in and out quickly without using the GUI buttons, simply hold down the **Ctrl** and **Shift** keys, and use the left mouse button to zoom in and the right one to zoom out. To find any item quickly, select it in the List window located in the upper-right corner of the GUI. Once you've located the desired item, hit F11. The Array window will zoom in on it.

FPGA Editor has four main windows: List, World, Array and Block. The List window shows all the active items in your design. The pulldown menu at the top of this window will allow you to select its contents—that is, a list of placed or unplaced components, nets or unrouted nets, and so on.

World's view window gives you a look at the complete FPGA die at all times; this comes in handy if you are trying to determine how you previously routed a net. The Array window, meanwhile, is your active view of the fabric and logic. When you double click on any item within the Array view, the Block view will appear, offering a detailed look at the item or logic element of interest.

You can duplicate any of these windows for easier navigation and editing of your design. In many cases it is handy to

FPGA Editor allows you to see your implemented design and review it to determine if that is truly what you wanted at the FPGA fabric level—a must for any engineer or FAE.

have a second Array window open to allow you to work on two different parts of the design at once. For example, let's say you had to add a route between a global clock buffer and a flip-flop at the bottom of the chip. It's much easier to do this if you have one Array window on the global clock buffer output and a second on the clock input of the flip-flop of interest. Otherwise, you will be zooming in and out to locate the source and destination of the route, which is not fun.

On the right side of the FPGA Editor GUI is a button bar with 20 function buttons to help you view and edit your design. You can add more function buttons with your own functions by editing the `fpga_editor.ini` file located in the `$XILINX/data` directory. While reviewing your design, use the **INFO** button from time to time. It dumps all the information on the selected item to the Console window. This can come in handy, since you can highlight the data in the Console window and copy it for use elsewhere, such as writing UCF constraints.

Once you have the basics down, you can start reviewing the fabric of the FPGA. I normally start my fabric review with the clocking logic. This would include the digital clock manager (DCM), phase-locked loop (PLL), global buffer (BUFG), regional clock buffer (BUFR), I/O buffer (BUFIO) and the different clock regions. (To alphabetize the items, go to the **LIST** window and click on the word **Type**.) Click on a **DCM** and hit **F11**. The **ARRAY** window will locate the selected DCM and zoom in on it. Go ahead and click on the **DCM** once and watch the Console win-

dow at the bottom of the GUI as it produces something like this:

```
comp "DCM_BASE_inst_star", site
"DCM_ADV_X0Y9", type = DCM_ADV
(RPM grid X73Y202)
```

This is useful data. **Copy** and **paste** the above line into your UCF and make the following changes to lock down this DCM logic:

```
INST "DCM_BASE_inst_star"
LOC=DCM_ADV_X0Y9;
```

Using this method, you can lock down almost anything in the FPGA. Here is another example for BUFG locking:

```
comp "BUFG_inst_star", site
"BUFGCTRL_X0Y20", type = BUFG
(RPM grid X73Y124)
```

```
INST "BUFG_inst_star"
LOC=BUFGCTRL_X0Y20;
```

Return to the List window again and highlight the same DCM. Double clicking on it will bring up the Block view of the DCM and show you all the settings and parameters. This is very powerful feature that can apply to any logic item within the fabric. If you select a slice and double click on it, you can see how that slice was routed and whether the carry chain or local flip-flop was used.

The Block view contains a button bar with many more options. One worth noting is the **F=** button, which displays the complete configuration of the items used in

that slice. For example, if you used a Lut6 and a flip-flop, the F= button would give you the Boolean equation for the LUT and the mode the flip-flop is configured for.

It is one thing to read the Xilinx user guide; it's another to see all the logic, switches and parameters spread out on your computer screen for review. Once you become familiar with where everything is located, you'll be surprised at how it will help you write and verify your design.

Scripting in Flow Patches

FPGA Editor has the ability to record your actions while you edit a design in the GUI. You can save and even play them back to reproduce your work at a later date. This is extremely powerful when it comes to making "in flow" changes to your design at times when it's not possible to change your RTL. Let's say you've created a design with third-party IP or Xilinx encrypted IP, and it contains a global clock and a DCM that generates a clock call `interface_clk`. Then let's assume the ASIC to which you're interfacing has a reported erratum and cannot accept data on the rising edge of the `interface_clk` as advertised. How do you fix this problem?

Well, you could alter your PCB and remove the broken ASIC or have the third-party IP team review altering the clock output logic to provide an `interface_clk` with a 90-degree phase shift. Both of these solutions are time-consuming and costly. A simpler suggestion would be to use FPGA Editor to record the actions and make the necessary changes to the `interface_clk` logic to provide the correct clock phase to the broken ASIC. Once you have an FPGA Editor script of the changes, you can play them back from your command line build script and continue your FPGA flow as normal. When the broken ASIC is fixed, you simply remove the FPGA Editor script playback from your build script and the `interface_clk` will return to its normal behavior.

To begin hand-editing your design, you need to enable read/write privileges in FPGA Editor. Go to the menu bar and click on **File** → **Main Properties**. Under this menu, you can adjust the edit mode

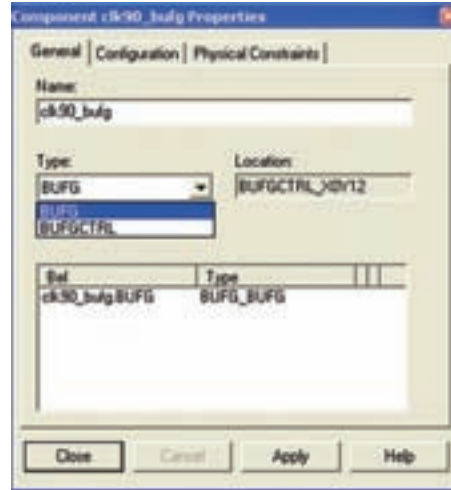


Figure 1 – The properties window allows the user to configure and name the selected logic item.

from **No logic change** to **Read/Write**. Click **Apply** and you can now edit your design. The next step is to begin recording all your changes with FPGA Editor; simply go to the menu bar and click on **Tools** → **Scripts** → **Begin Recording**. FPGA Editor will prompt you for a script name (such as `patch.scr`). Once you've entered it, you can begin making the necessary changes to your design.

It is always a good idea to run a design rules check (DRC) on your design to see if

it raises any red flags. In my example design, I have 14 warnings that should be ignored. Next we will need to locate the DCM for the `interface_clk` and create another clock called `DCM_clk90_out` from that DCM's 90-degree output. You will need to route that clock to a BUFG to use the global clock routing. To add a BUFG, simply find an unused BUFG location in the fabric, right-click on it and select **Add**. The tool will then prompt you to give the BUFG a name (`clk90_bufg`) and determine its type: BUFG (see Figure 1).

Once you've created the new BUFG, you need to hook up its input and output to the desired locations. In this case, the DCM's 90-degree output will drive the BUFG. To make this connection, in window `Array1`, click on the DCM's 90-degree output pad and in window `Array2`, click on the input pad of the BUFG while holding down the **Ctrl** key. Then release the **Ctrl** key, click your right mouse button and select **Add**. The tool will prompt you for a name of that new net connection. This in turn links the DCM and BUFG together via the new net (see Figure 2).

The output of the `clk90_bufg` needs to replace the clock on an IOB that is driven by the original `interface_clk`. To remove

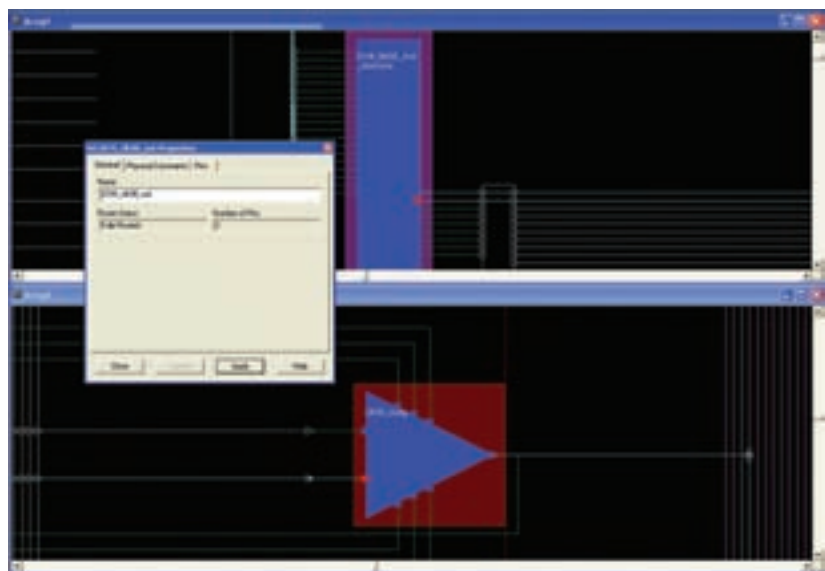


Figure 2 – When hand routing between two logic items, use two Array windows for easy selection of the source and destination, as shown by the red triangles.

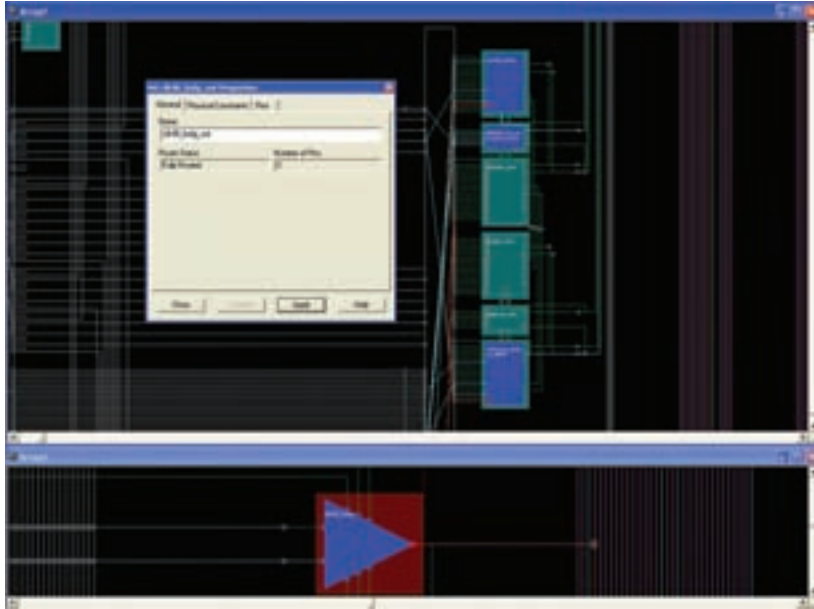


Figure 3 – The BUFG output net properties window shows the number of net connections and the fully routed net status.

the IOB from the original clock domain, you need to locate the IOB, highlight the clock input pads and hit the **Delete** key to remove this connection. This will in turn allow us to connect the clock from our new `clk90_bufg` and complete the patch. To connect the output of the BUFG (`clk90_bufg`), highlight the output pad of the BUFG in window `Array2` and select the clock inputs of the IOB in window `Array1` while holding the **Ctrl** key. Release the **Ctrl** key and click the right mouse button to display the option menu and select **Add**. This makes the final connection between the BUFG output and the IOB that drives the newly created interface to the downstream ASIC, which in turn allows `interface_clk90` to capture the transmitted data correctly.

This completes the patch for the ASIC. Now you should rerun the DRC checker to make sure you didn't introduce any new errors. Go to the menu bar and click on **Tools** → **DRC** → **Run**.

Once your script is complete and error free, you need to go back to the menu bar and select **Tool** → **Script** → **End Recording**. This will stop and close the script for use the next time you wish to make any RTL changes and this ASIC patch is required. It's a good idea to open

the script file in a text editor and remove all the GUI **Post** and **Unpost** commands. These commands are not needed, and they make the script hard to read and review. The text below is the script for our ASIC patch. As you can see, it is fairly straightforward and easy to read.

```
unselect -all
setattr main edit-mode Read-Write
add -s "BUFGCTRL_X0Y28" comp
clk90_bufg ;
setattr comp clk90_bufg type BUFG
unselect -all
select pin 'BUFGCTRL_X0Y28.IO'
select pin 'DCM_ADV_X0Y11.CLK90'
add
post attr net $NET_0
setattr net $NET_0 name
DCM_clk90_out
unselect -all
select pin 'OLOGIC_X0Y2.CLK'
delete
unselect -all
select pin 'ILOGIC_X0Y3.CLK'
delete
unselect -all
select pin 'ILOGIC_X0Y3.CLK'
```

```
select pin 'OLOGIC_X0Y2.CLK'
select pin 'BUFGCTRL_X0Y28.O'
add
post attr net $NET_1
setattr net $NET_1 name
clk90_bufg_out
unselect -all
drc
save -w design "patch.ncd"
"patch.pcf"
exit
end
```

Take a look at the script and see if you can pick out the actions you did in the GUI.

It's important to understand that you can play back this script from the GUI (under the menu bar **Tool** → **Scripts** → **Playback**) or the command line. To play back your patch from your build script, simply add the following command:

```
fpga_edline yourdesign.ncd
yourdesign.pcf -p yourscript.scr
```

You should execute this command after PAR, when the NCD and PCF files are completed.

FPGA Editor truly is a power-user tool, although not everyone would want or need to use it in their designs. But when you need something special or you need to bend the rules a bit to get even more from your design, there is no other tool like it. Your FAE is the one who can show it to you, and demonstrate how FPGA Editor can help you with debug, verification and, of course, bending the rules. 🌈

Clayton Cameron is a Senior FAE based in Toronto. He joined Xilinx in 2000, supporting telecom customers in the Ottawa office. As an FAE, Clayton greatly enjoys helping customers and solving problems. He also enjoys the diversity of his position and the variety of challenges he faces on a daily basis.

In his spare time, he lets off steam in the gym, keeping physically and mentally fit. At home, he loves to spend time with his wife and two young children..