Maximizing FPGA Design Performance Using Amplify from Synplicity

In very large designs, interconnect delays are becoming more and more predominant. Amplify™ is the first physical synthesis tool for FPGAs that helps you optimize signal routing delays and achieve timing closure.

by Philippe Garrault
Technical Marketing Engineer, Xilinx, Inc.
philippe.garrault@xilinx.com

Amplify is a new product from Synplicity® that allows you to add physical constraints to your design, using the Synplify® PRO synthesis engine. It links your RTL code to a floorplan of the target device. You can assign your RTL code to physical regions by graphically dropping the logic into these regions. During the synthesis process, Amplify will use a set of rules to optimize the RTL code based on estimated placement and interconnect delays (into and between regions).

Amplify generates an optimized netlist (.edf file) and a Xilinx constraint file (.ncf file) based on the specified physical constraints. These files are then used by the Xilinx Alliance Series 3.1i software tools to implement the design.

Improving Maximum Frequency

With Amplify you can iterate the implementation twice. In the first pass, you determine the critical paths after place and route. In the second pass, you assign physical constraints on these critical paths to optimize the netlist. By iterating the second pass, the physical constraints can be refined according to the place and route post-layout timing report, until your timing requirements are met.

Another approach is to set several regions prior to implementation. These regions could be a representation of the RTL hierarchy of the design and each could contain one block (or module) of the design.

Finally, you could also mix these techniques to get a floorplan that would not only be a hierarchy representation but you could also add physical constraints on the critical paths regardless of any hierarchy consideration. This is what we did in the following example.

Design Example

The following example shows how to interface Amplify with the Xilinx place and route tools. The design presented is a network application, which is divided into a top module driving nine similar sub-modules. This code is implementing FIFOs and large busses. We targeted a Virtex-E, XCVE1000-7 device.

On the first pass, the project is synthesized then implemented with global timing constraints only (without physical constraints). Figure 1 shows a floorplanned view of the design after the first pass. Note that the logic is spread over the chip because, without placement constraints, the place and route algorithm has no information about what logic is crucial for grouping into a region (or there are too many possibilities). The timing constraints were not met, and the best frequency obtained was 104.6MHz.

After this first implementation, we analyzed the post-layout timing report to gather information on the critical path, such as:

- The number of critical paths.
- The start and end points (single or multiple).
- The type of critical path (link with I/O or purely internal, wire or bus).
- The number of logic levels.
- The device resources (flip-flop, combinatorial, block RAM, and so on).

On the second pass, the different critical paths were assigned in separate regions through the Amplify user interface. Figure 2 shows the physical constraints entered in the synthesis environment for our example.

The regions are floorplanned according to the required design resources, such as block RAM and high fanout nets. . By re-synthesizing the design with these physical constraints, a new netlist file along with a constraint file were created. The place and route software uses these optimized files to constrain the logic on these particular areas by placing the critical logic together, shortening net delays to meet the timing requirements.

If the constraints are not met, the Xilinx floorplanner can give useful information about the final layout and determine the precise logic utilization within a region, or view the exact placement of logic on the critical path. This can help to resize regions, remove constraints on non-critical paths, or re-place regions closer together to drive or share common logic or busses.

Figure 3 shows the floorplanned view of the constrained design in our example. Note that the logic is gathered according to the constraint file. The best frequency obtained was 120.1MHz, which is a 12.9% improvement. Using the post-layout timing report and the floorplanner helped us focus on the critical parts of the design, thus saving time by applying more accurate constraints, and reducing the number of iteration needed to meet the timing constraints.

Interfacing Amplify with Xilinx Tools

Here are some guidelines to help you get the best performance from these tools:

- Put different critical paths into different regions. This usually gives better results.
- If the critical path contains lots of logic, two regions can be overlapped: one small one containing the most critical logic that needs to be placed very close together and a bigger one containing the rest of the critical path.
- Amplify does not currently write constraints for block RAM or black boxes, thus if the critical path includes these objects, use the Constraints Editor and constrain black-boxes and block RAM in the UCF file within or close to the region constraining the rest of the critical path. Use the following command:

```
INST p1.qram1 LOC = RAMB4_R0C1:
RAMB4_R7C1, RAMB4_R*C2;
```

- Applying physical constraints to critical paths is more likely to improve results if the ratio between routing and logic is in favor of routing (this ratio is given by the post-layout timing report).
- Slightly moving a region can affect the maximum frequency of your design.

Conclusion

Synplicity's Amplify Physical Optimizer in conjunction with the Xilinx Alliance Series 3.1i software can significantly improve your design performance. As a result of the new features and capabilities, you have a more efficient way to visualize and constrain critical paths of your designs, saving time in multiple iterations while getting better speed performance.

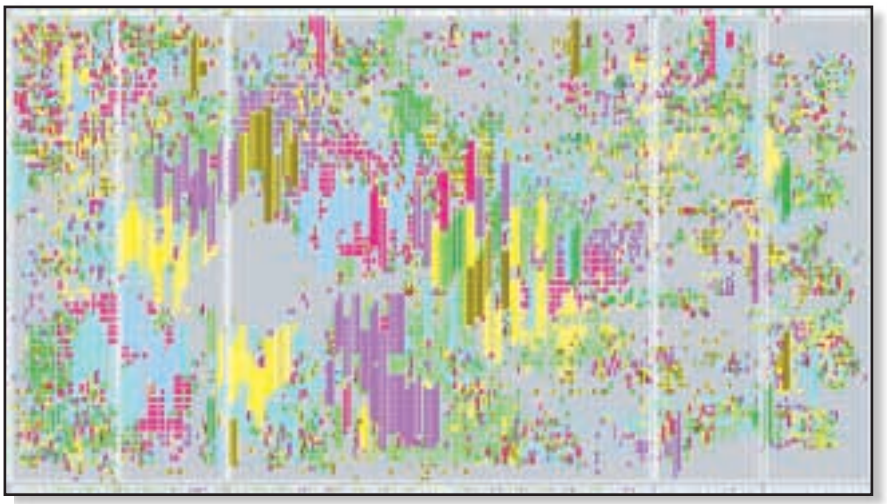

Figure 1 - floorplanned view of the design on the first pass.

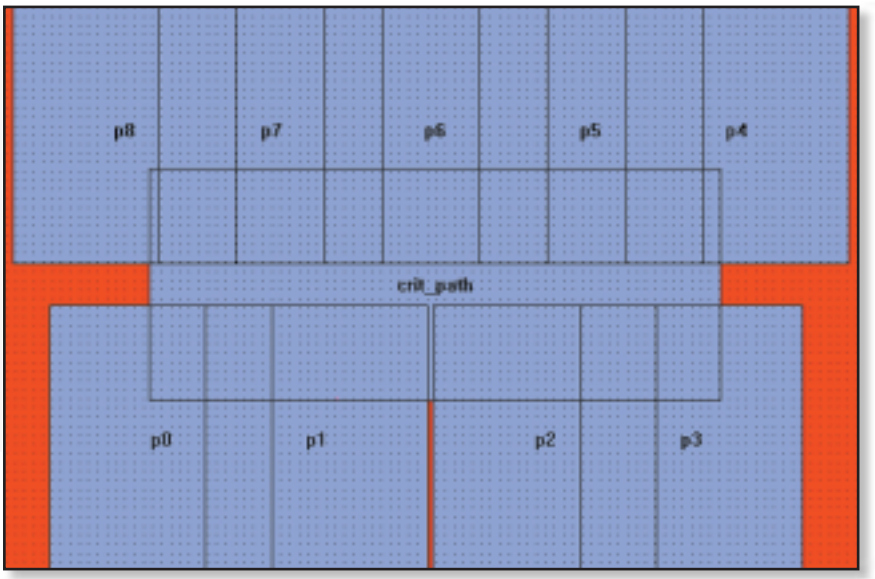

Figure 2 - Physical constraints entered in the synthesis environment.

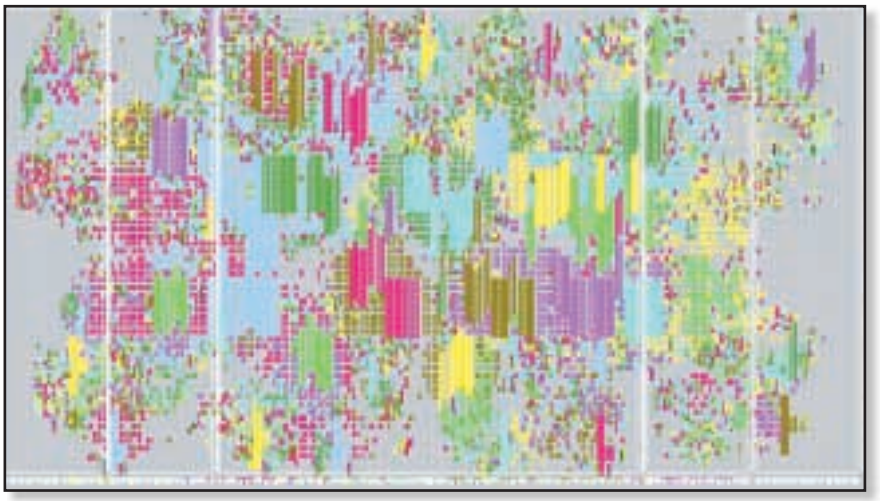

Figure 3 - Floorplanned view of the constrained design.