XILINX®

XAPP422 (v2.0) March 10, 2004

Creating RPMs Using 6.2i Floorplanner

Summary

Relationally Placed Macros (RPMs) are frequently used in designs that have predefined modules or specific elements that need to be placed in such a way as to get highly predictable timing and performance. Floorplanner is a GUI-based tool that allows one to view and make these RPMs through the MacroBuilder capability. This application note explains the steps to create, instantiate, and implement a design with RPMs that were created in Floorplanner.

Introduction

The ability to group elements in a certain pattern is a powerful technique one can employ in their designs in order to meet timing objectives. Xilinx has been using this approach in their cores by creating RPMs. This application note takes you through the steps to introduce and familiarize you with the MacroBuilder and instantiation of RPMs in a larger design.

Flow of Creating a RPM

The two main functions of the MacroBuilder are to create the RPM from an existing file using Floorplanner and to instantiate the RPM in a project. The creation of the RPM can be done after NGDBuild or after PAR. The RPM can be instantiated multiple times in the project and can be verified using PACE, FPGA Editor, or Floorplanner.

Creating RPM After NGDBuild

The creation of the RPM, after NGDBuild, is based upon the HDL source code. During synthesis, make sure that the *I/O insertion* and the *Creation of I/O Pads from Ports* are disabled. After synthesis, launch Floorplanner to create the RPM. This can be done from command line (floorplanner) or from ISE, double-click Floorplan Design, which is in the Process Window, under the Implement Design folder and under the Translate folder.

Once Floorplanner is open, expand the RPM design in the hierarchy window and start placing the components via drag and drop to the Editable window. After you have placed all of the components in the shape you want, then select *File -> Write RPM to UCF.* This brings up a dialogue box to create the RPM. Please specify the UCF file that you wish the constraints to be written to.

In addition to creating a UCF for the RPM, you can check the *"Automatically generate RPM macro in NGC format"* box to have MacroBuilder combine the original netlist and the new RPM UCF to create an RPM NGC file. Xilinx recommends using this option and to name the RPM NGC file a different name than the original netlist. Since the NGC file contains all of the RPM design and the RLOC constraints, the NGC file is the only required file to instantiate the RPM into the project or larger design.

To use the RPM NGC file in the project, you can either set the Macro Search Path for NGDBuild or copy the file into the current project directory. If you are doing many RPM cores, Xilinx recommends creating a "rpm_core" directory to store all of the cores.

Creating RPM After PAR

The creation of the RPM, after PAR, is based upon the netlist files that are pre-synthesized with the same options disabled as in the previous RPM creation. In addition to those options being disabled, the Create I/O Pads from Ports and Trim Unconnected Signals need to be disabled in NGDBuild (Translate) and MAP, respectfully. After Place and Route, launch Floorplanner to create the RPM. This can be done from command line (floorplanner) or from ISE. Double-click Floorplan Design, which is in the Process Window, under the Implement Design folder and under the Place and Route folder.

Once Floorplanner is open, select Floorplan -> Replace All with Placement to place the logic in the editable window in the same shape that came from Place and Route. If you do not wish for all of the components to be in the RPM, you can remove them from the Editable window. Since all of the components for the RPM are placed, select File -> Write RPM to UCF. This brings up a dialogue box to create the RPM. Please specify the UCF file that you wish the constraints to be written to.

In addition to creating a UCF for the RPM, you can check the "Automatically generate RPM macro in NGC format" box to have MacroBuilder combine the original netlist and the new RPM UCF to create a RPM NGC file. Xilinx recommends that you use this option and name the RPM NGC file a different name than the original netlist. Since the NGC file contains all of the RPM design and the RLOC constraints, the NGC file is the only required file to instantiate the RPM into the project or larger design.

To use the RPM NGC file in the project, you can either set the Macro Search Path for NGDBuild or copy the file to the current project directory. If you are doing many RPM cores, Xilinx recommends creating a "rpm_core" directory to store all of the cores.

Instantiating the RPM

The instantiation of the RPM NGC file is very easy. During synthesis of the project design, make sure the *Add I/O Buffers* and the *Create I/O Pads from Ports* are enabled. Also, during implementation, make sure that the *Create I/O Pads from Ports* and *Trim Unconnnected Signals* are enabled in NGDBuild (Translate) and MAP, respectfully. The RPM NGC file is instantiated as a "black box" in the HDL code. Make sure that the instance name and ports match the NGC file. The project design is now ready to be implemented with as many RPMs as you wish.

Once Place and Route is complete, you can verify the placement of the RPM in Floorplanner by double-clicking View/Edit Placed Design (FloorPlanner) under the Place and Route folder.

Conclusion

The result of the RPM creation flow is a user-defined "core" that can be instantiated into a larger, top-level design. The core is comprised of the original synthesized netlist combined with the Floorplanner-generated UCF file in the RPM NGC file.

For relatively small RPMs, the user can directly create the RPM manually by placing the design symbols in Floorplanner after NGDBuild (Translate). For larger RPMs, it might be more efficient to implement the design through Place and Route and use the post-PAR design as the basis for the RPM.

Once the RPM is created, it is instantiated as a black box in the HDL code and implemented through the Xilinx tools.

After reading this application note, you should have a good understanding of how to use the Floorplanner's MacroBuilder capability to generate and use RPMs. For RPMs containing Block RAMs or Multipliers, RPM_GRID should be used to keep the relative placement between SLICE logic and BLOCK RAM or Multipliers. Application Note XAPP416 has more detail description on implementation of RPM_GRM.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/02/02	1.0	Initial Xilinx release.
3/10/04	2.0	Revised to Creating RPMs in 6.2i Floorplanner