XILINX®

XAPP422 (v1.0) December 3, 2002

Using 5.1i Floorplanner to Create RPMs

Summary

Relationally Placed Macros, RPMs, are used frequently in designs that have predefined modules or specific elements need to be placed in such a way to get highly predictable timing. Floorplanner is a GUI-based tool that allows one to view their design and make changes to the placement. In Foundation™ ISE 5.1i, Xilinx has provided the option of using Floorplanner's MacroBuilder capability to create RPMs from smaller designs and use them as building blocks. This application note explains the steps to create, instantiate, and implement a design with an RPM that was 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 two labs to introduce and familiarize yourself with the new feature in Floorplanner. It also illustrates some current issues with the process and presents workarounds. Currently, XST has limited support for this new feature such that it can only support this flow with designs that instantiates an existing pre-synthesized EDN or EDF module. The labs were created using Synplify to synthesize the designs. The first lab is a basic scenario where you create an RPM from an existing design and instantiate it one time in a larger design. The second lab is the same as the first except it will instantiate the RPM more than once.

Lab One

This lab describes the flow of creating an RPM of an 8-bit cascaded counter (CB8CE) and instantiating the RPM in top.vhd. The **lab1.zip** (http://www.xilinx.com/xapp/lab1.zip) reference design file can be used for this lab.

1. Creating RPM from an existing file using Floorplanner
    a. NGDBUILD Flow: With Synplify as synthesis tool.
    b. POST PAR Flow: Using just EDN as input source.
2. Instantiating the RPM in a project.
3. Verifying the RPM component has been placed correctly.

This lab requires the following tools.

1. ISE 5.1i .
2. Synplify 7.1 (Netlist files are included, if Synplify is not available).
3. Win2000/NT.

RPM Generation (NGDBUILD FLOW with Synthesis)

This section of the lab will go over creating an RPM from source code. To begin this lab, unzip lab1.zip to a working directory. Since this part of the lab requires Synplify, simply go through the flow and if it is not available then jump to the next section.

- Start Project Navigator 5.1i and open rpm_cb8ce.npl
- Make sure the design flow is Synplify VHDL.[1] If not, change the design flow by right clicking on the device line in the Source window and select properties. Change the design flow to Synplify VHDL.
- Double click rpm_counter_top.vhd in the source window to see the design code structure.
- Right click on Synthesize and select properties
- Select the Device Options tab and Disable *I/O insertion*.
- Right click on Translate and select properties. Uncheck *Create I/O Pads from Ports*.
- Double click on Floorplan Design under Translate of the Implement Design in the Process window.
- Expand counter in the hierarchy window. (See Figure 1)

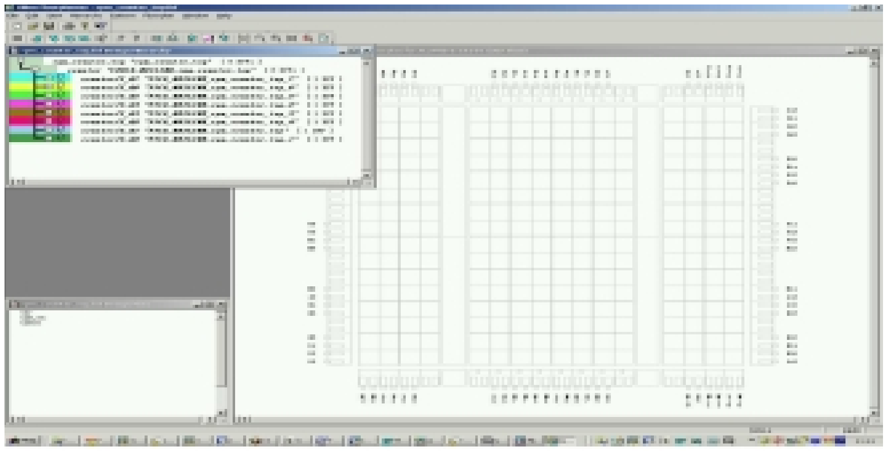

Figure 1: **Expanded Counter Module in Hierarchy Window**

- Start placing the DFF components via drag and drop to the editable window. (See

1. Using Synplify as synthesis tool is required because of a limitation with NGDBUILD which requires EDN/EDF file in order for NCF file to be read in. XST produces NGC formatted netlists, not EDIF, which will not be read in with NCF file by NGDBUILD. This limitation will be addressed in a future release of ISE.

Figure 2)

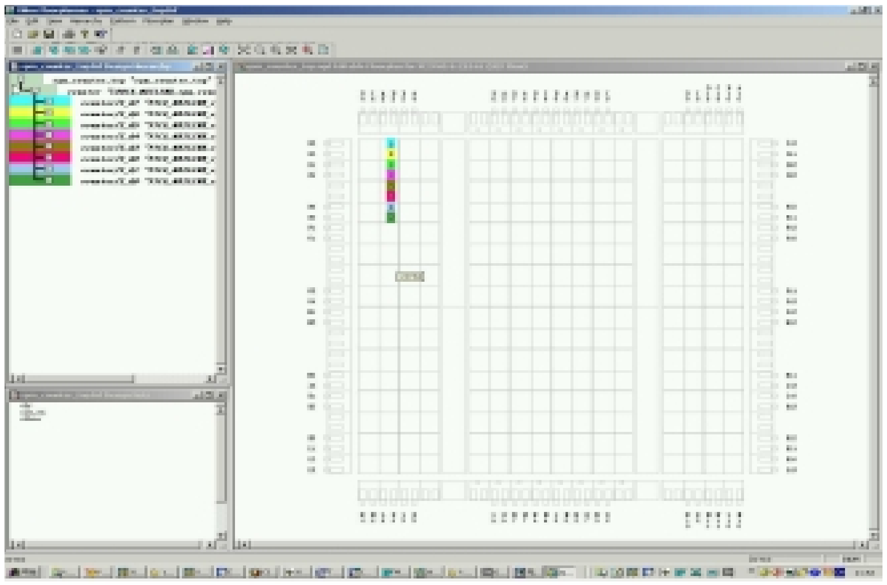

Figure 2: **Placed Logic**

- Click on *File -> Write RPM to NCF*.
- Exit Floorplanner without saving.
- Browse to the NCF (rpm_counter_top.ncf) file in the project folder with Windows Explorer and open it with Wordpad. It should contain the following constraints.

```
# Start of RPM Constraints extracted by Floorplanner from the Design
INST "counter/I_Q7/I_36_35" RLOC = "X1Y15" ;
INST "counter/I_Q7/I_36_35" U_SET = "rpm_counter_top" ;
INST "counter/I_Q6/I_36_35" RLOC = "X1Y15" ;
INST "counter/I_Q6/I_36_35" U_SET = "rpm_counter_top" ;
INST "counter/I_Q5/I_36_35" RLOC = "X1Y14" ;
INST "counter/I_Q5/I_36_35" U_SET = "rpm_counter_top" ;
INST "counter/I_Q4/I_36_35" RLOC = "X1Y14" ;
INST "counter/I_Q4/I_36_35" U_SET = "rpm_counter_top" ;
INST "counter/I_Q3/I_36_35" RLOC = "X1Y13" ;
INST "counter/I_Q3/I_36_35" U_SET = "rpm_counter_top" ;
INST "counter/I_Q2/I_36_35" RLOC = "X1Y13" ;
INST "counter/I_Q2/I_36_35" U_SET = "rpm_counter_top" ;
INST "counter/I_Q1/I_36_35" RLOC = "X1Y12" ;
INST "counter/I_Q1/I_36_35" U_SET = "rpm_counter_top" ;
INST "counter/I_Q0/I_36_35" RLOC = "X1Y12" ;
INST "counter/I_Q0/I_36_35" U_SET = "rpm_counter_top" ;
```

- Move rpm_counter_top.edn and rpm_counter_top.ncf into the pre-made rpm_core folder using Windows Explorer.

RPM Generation (POST PAR FLOW without Synthesis)

This section goes through creating the RPM with pre-synthesized netlist files[1].

- Start Project Navigator 5.1i and open rpm_cb8ce.npl
- Remove test_counter.vhd from the source window and change the design flow to EDIF by right clicking on the device line in the Source window and select properties. Change the design flow to EDIF.

1. To correctly use any netlist as a RPM core, the netlist must not have IO buffer inserted.

- Add rpm_counter_top.edn from RPM_FILE folder to the project. (Project -> Add Copy of Source)[1]
- Right click on Translate and select properties. Uncheck *Create I/O Pads from Ports*
- Right click on Map and select properties. Uncheck *Trim Unconnected Signals*.
- Select the source, rpm_counter_top.edn, in the source window and double click on *View/Edit Placed Design (Floorplanner)* under *Place & Route* of the Implement Design in the Process Window.
- Click on *Floorplan -> Replace All with Placement* to place logic in the editable window (Figure 3).

1. Make sure add copy of source is used because if add source is selected rather than add copy of source, project navigator will pick up undesired constraints file existing in the rpm_file folder.

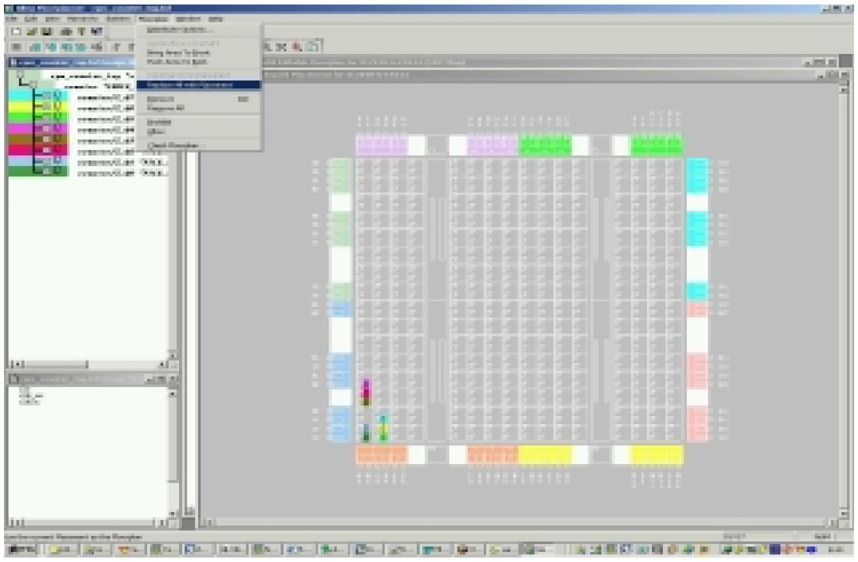

Figure 3: **Implemented Logic Placement View After PAR**

- Click on File -> Write RPM to NCF and exit Floorplanner.
- Check the NCF (rpm_counter_top.ncf) file in the project folder with windows explorer and it should contain the following constraints.

```
# Start of RPM Constraints extracted by Floorplanner from the Design
INST "counter/I_Q7/I_36_35" RLOC = "X1Y1" ;
INST "counter/I_Q7/I_36_35" U_SET = "rpm_counter_top" ;
INST "counter/I_Q6/I_36_35" RLOC = "X1Y0" ;
INST "counter/I_Q6/I_36_35" U_SET = "rpm_counter_top" ;
INST "counter/I_Q5/I_36_35" RLOC = "X1Y0" ;
INST "counter/I_Q5/I_36_35" U_SET = "rpm_counter_top" ;
INST "counter/I_Q4/I_36_35" RLOC = "X0Y3" ;
INST "counter/I_Q4/I_36_35" U_SET = "rpm_counter_top" ;
INST "counter/I_Q3/I_36_35" RLOC = "X0Y2" ;
INST "counter/I_Q3/I_36_35" U_SET = "rpm_counter_top" ;
INST "counter/I_Q2/I_36_35" RLOC = "X0Y2" ;
INST "counter/I_Q2/I_36_35" U_SET = "rpm_counter_top" ;
INST "counter/I_Q1/I_36_35" RLOC = "X0Y0" ;
INST "counter/I_Q1/I_36_35" U_SET = "rpm_counter_top" ;
INST "counter/I_Q0/I_36_35" RLOC = "X0Y0" ;
INST "counter/I_Q0/I_36_35" U_SET = "rpm_counter_top" ;
```

- Move the netlist rpm_counter_top.edn, and the rpm_counter_top.ncf into the pre-made rpm_core folder with Windows Explorer.

Instantiation of RPM

This section will go through instantiating the RPM files in the project.

- Continuing from the previous section remove the source file, rpm_counter_top.vhd (NGDBUILD flow, Figure 4) or rpm_couter_top.edn (Post PAR flow, Figure 5), from the project.
- Change the design flow to VHDL if continued from Post PAR flow by right clicking on the device line in the Source window and select properties. Change the design flow to XST VHDL.
- Add the VHDL source file, top.vhd.
- Make sure *Add I/O Buffers* of the *Xilinx Specific Options* in **synthesis properties** is enabled if XST VHDL is used. If Synplify VHDL is still used, make sure *Disable I/O Insertion* is unchecked under synthesis properties.

- Set the *Translate* properties to search for the macro in the rpm_core folder and make sure *Create I/O Pads from Ports* box is checked.
- Make sure *Trim Unconnected Signals* in Map properties is checked if continuing from Post Par Flow.
- Double click on *View/Edit Placed Design (FloorPlanner)* under *Place & Route* to implement the design and launch Floorplanner to verify the RPM has been correctly implemented.

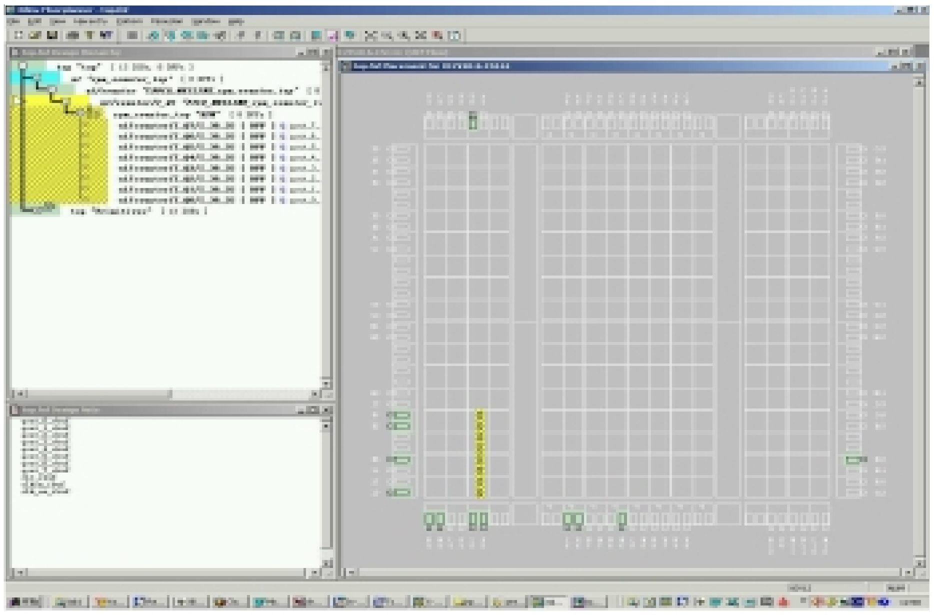

Figure 4: **Placed RPM View in Floorplanner (NGDBUILD Flow)**

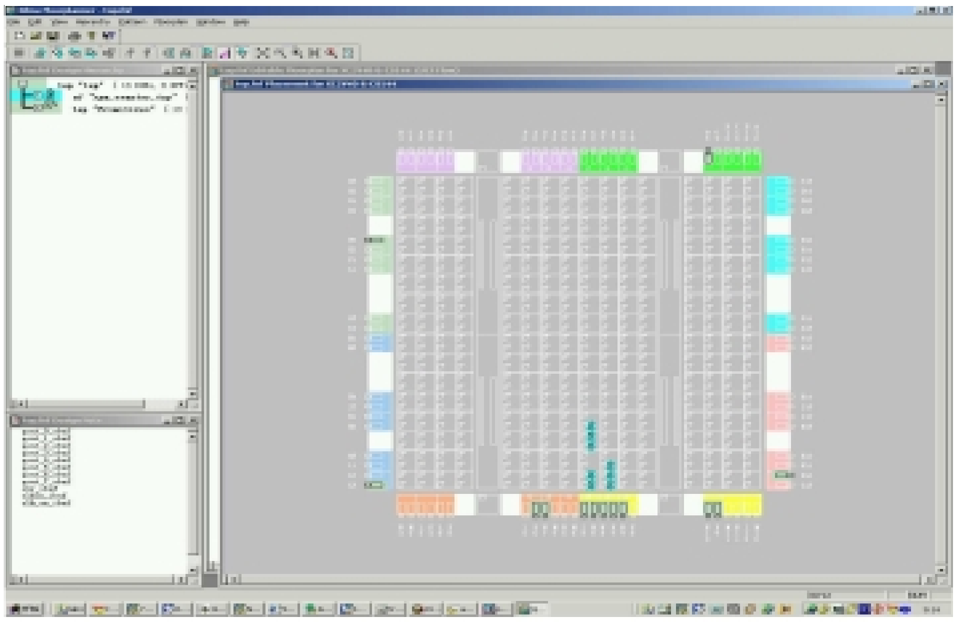

Figure 5: **Placed RPM View in Floorplanner (Post Par Flow)**

Lab One Conclusion

The result of the RPM creation flow is a user defined “core” that can then be instatiated into a larger, top-level design. This core is comprised of the combination of the original synthesized EDIF netlist and the Floorplanner-generated NCF file.

For relatively small RPMs, user can directly enter the Floorplanner after the Translate process (NGDBUILD) and manually place all of the design symbols.

For larger RPMs, it may be more efficient to implement the design through PAR and use the post-PAR design (placed NCD file) as the basis for the RPM. In this case, you will need to disable map trimming operations via either the “-u” command line switch or unchecking the “Trim unconnected signals” item in the Map properties dialog box in ISE. Implement the design through PAR, and then use the Floorplanner to read in both the NGD and placed NCD file. You can then use the “Floorplan -> Replace All With Placement” menu item to impose the placed design onto the Editable Floorplan view. The remainder of the RPM creation process is identical to that described in the prior section of this lab.

Lab Two

This lab will illustrate instantiating two instances of an RPM. In 5.1i, there are some limitations for top-level designs that instantiate multiple copies of a Floorplanner generated RPM. These limitations will be explained in this lab, and workarounds will be presented. The **lab2.zip** (http://www.xilinx.com/xapp/lab2.zip) reference design file can be used for this lab.

1. Implementing the design with two instantiations of an RPM to discover the limitations of this flow.
2. Modifying the RPM files to workaround the limitation.
   a. Workaround for RPM with hierarchical structure via “uniquification”.
   b. Workaround for RPM without hierarchical structure.
   c. Workaround by modifying ucf file associated with top-level source code.
   d. Workaround by augmenting the NCF constraints.

Implementation of a Top-Level Design Containing Two Instances of an RPM

- Unzip lab2.zip into any working directory and open rpm_cb8ce.npl.
- Select top_hier from the source window.
- Right click on *Translate* and select properties. Make sure the Macro search path is pointing to the RPM_CORE folder.
- Double click on *Implement Design in the Process* window; MAP should quit with the following error. [1]

```
Design Summary
--------------
Number of errors : 4
Number of warnings : 0

Section 1 - Errors
------------------
ERROR:Pack:679 - Unable to obey design constraints
(MACRONAME=rpm_counter_top,
 RLOC=X1Y12) which require the combination of the following symbols into a
 single SLICE component:
  FLOP symbol "hier_core1/counter/I_Q1/I_36_35" (Output Signal =
qout_1_obuf)
  FLOP symbol "hier_core2/counter/I_Q0/I_36_35" (Output Signal =
qout1_0_obuf)
  FLOP symbol "hier_core1/counter/I_Q0/I_36_35" (Output Signal =
qout_0_obuf)
```

1. The Map error is due to the duplicate U_SET name used for two different RPM instantiations.

```
  FLOP symbol "hier_core2/counter/I_Q1/I_36_35" (Output Signal =
qout1_1_obuf)
 Symbols have different XGROUP parameters. Please correct the design
 constraints accordingly.
```

Modification of the Project

A. Workaround for RPMs with Hierarchical Structure via “Uniquification”

This section goes over working around the limitation of instantiating multiple RPMs. This workaround effectively uniquifies each instance of the RPM. This involves making each RPM have a different name and separate U_SET properties in the NCF file.

- Open top_hier.vhd and uncomment lines 38-45 and remove “- -“ in line 59.
- Browse to the folder rpm_core where there are two EDN and two NCF files. (rpm_counter_top.edn/ncf , rpm_counter_top1.edn/ncf)
  - The EDN files are identical except that they differ only by name.
  - The NCF files are essentially the same, except rpm_counter_top1.ncf has modified U_SET name.
- Save the file and re-implement the project.
- The project should now implement correctly. See Figure 6 for to view the final implementation in Floorplanner.
- **Note:** Although removing the U_SET lines in a hierarchical RPM would help the design pass MAP, the implemented result will not keep the RPM shape — each hierarchical level in the “core” design gets split out into their own separate RPM. The U_SET keeps them all part of the same RPM. If U_SET is modified to be HU_SET, the design would also “pass” MAP but the RPM shape will not be kept either.

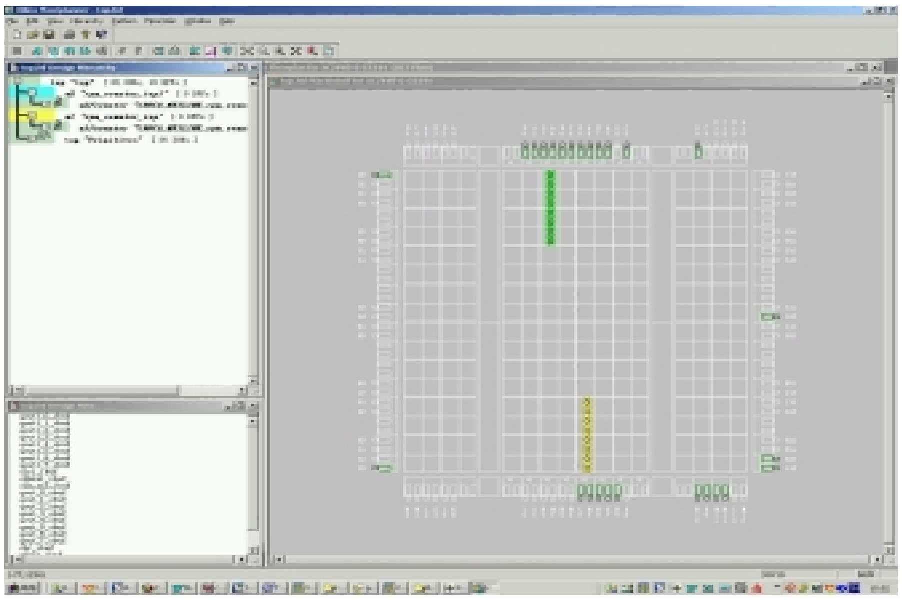

Figure 6: **Implemented RPMs in FloorPlanner**

B. Workaround for RPMs with Flattened Structure

RPMs that do not have hierarchy, i.e., they are “flat”, can be instantiated multiple times without uniquifying the netlist and NCF files. This section goes through how to work around the multiple RPM instantiation problem with a flattened RPM netlist.

- With lab2 still opened in Project Navigator, select top_flat.vhd from the source window.
- top_flat.vhd has the same code as top_hier except top_flat.vhd instantiates a flattened

version: rpm_counter_top_flat.edn[1].

- Right click on *Translate in the Process* Window, select Properties and make sure the macro search path is still pointing to the RPM_CORE folder.
- Browse to the RPM_CORE folder and open up rpm_counter_top_flat.ncf with Wordpad.

Comment out all the lines containing U_SET with # and save the file[2].

```
# Start of RPM Constraints extracted by Floorplanner from the Design
INST "counter_I_Q7_I_36_35" RLOC = "X1Y15" ;
#INST "counter_I_Q7_I_36_35" U_SET = "rpm_counter_top" ;
INST "counter_I_Q6_I_36_35" RLOC = "X1Y15" ;
#INST "counter_I_Q6_I_36_35" U_SET = "rpm_counter_top" ;
INST "counter_I_Q5_I_36_35" RLOC = "X1Y14" ;
#INST "counter_I_Q5_I_36_35" U_SET = "rpm_counter_top" ;
INST "counter_I_Q4_I_36_35" RLOC = "X1Y14" ;
#INST "counter_I_Q4_I_36_35" U_SET = "rpm_counter_top" ;
INST "counter_I_Q3_I_36_35" RLOC = "X1Y13" ;
#INST "counter_I_Q3_I_36_35" U_SET = "rpm_counter_top" ;
INST "counter_I_Q2_I_36_35" RLOC = "X1Y13" ;
#INST "counter_I_Q2_I_36_35" U_SET = "rpm_counter_top" ;
INST "counter_I_Q1_I_36_35" RLOC = "X1Y12" ;
#INST "counter_I_Q1_I_36_35" U_SET = "rpm_counter_top" ;
INST "counter_I_Q0_I_36_35" RLOC = "X1Y12" ;
#INST "counter_I_Q0_I_36_35" U_SET = "rpm_counter_top" ;
```

- Double click on *View/Edit Placed Design (Floorplanner)* under *PAR* in the process window to verify the design has been implemented correctly. See Figure 6 for verification.

C. Workaround by Modifying Constraints File

This work around uses RLOC constraints in the top-level design's UCF file. This method effectively makes the two instances of the counter RPM into a single, larger instance. The downside of this approach is that the two RPMs can no longer be placed independently by PAR. Also, this approach gets much more complex as the number of instantiations increases.

- Unzip lab2.zip to a different folder to get the original files.
- Open rpm_cb8ce.npl.
- Right click in the source window and select Add **Copy** of Source. Browse to rpm_files folder and add both top.ucf and top.vhd.
- Upon adding the UCF file, a pop up window will inquire about which file to associate top.ucf with. Select top to be associated with top.ucf file.

With the top.vhd file selected in the Source Window, double click on *Edit Constraints (Text)* under *User Constraints* in the Process Window to view the already modified constraint file[3].

```
INST U1/flat_core1 RLOC = X0Y0;

INST U2/hier_core1 RLOC = X2Y0;

INST U1/flat_core2 RLOC = X1Y0;

INST U2/hier_core2 RLOC = X3Y0;
```

- Right click on *Synthesize* in the Process Window and select *Properties*. Check *Keep*

1. Most synthesis tool allows generation of flatten netlist file. Please consult the software manual of the synthesis tool to see how this would be accomplished.
2. Since Project Navigator does not detect modifications to the NCF file, re-implementation steps after Translate is necessary.
3. The hierarchy to the component path is set for XST. For Leonardo Spectrum or Synplify, use "/" to represent hierarchy.

Example: INST U1/flat_core1 RLOC = X0Y0; When running this lab with Synplify7.0 as synthesis tool, remove the commented line on line 59 in top.hier.vhd. Synplify has a bug recognizing "- -1" as comment.

Hierarchy box under *Synthesis Options*.

- Right click on *Translate* in the Process Window. select *Properties* and make sure the macro search path is pointing to the RPM_CORE folder.
- Implement Design and double click on *View/Edit Placed Design (Floorplanner)* under *PAR* in the process window to verify the design has been implemented correctly (Figure 7).

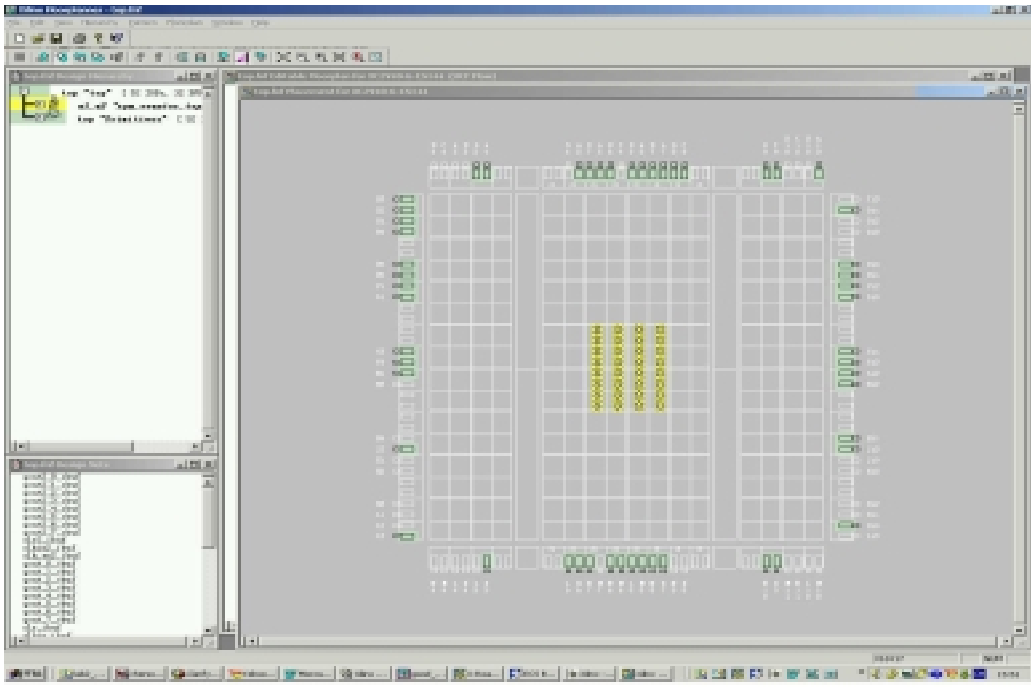

Figure 7: **Implemented View of Multiple RPM Instantiation by Modifying UCF File**

D. Workaround by Augmenting the NCF Constraints

This workaround involves adding additional constraints to the Floorplanner generated NCF file. The addition of “normalizing” RLOC constraints to the each intermediate level of hierarchy results in the whole function being automatically unified into a single RPM structure. To come up with the intermediate hierarchical names, a “shortcut” will be employed: map will automatically report these names in its report file where they can then be “cut-and-pasted” into the NCF file.

- Unzip lab2.zip to any working directory.
- Double click on top_level.vhd in the Source Window to see the design structure.
- Right click on *Translate* in the Process Window and select *Properties*. Make sure the macro search path is pointing to rpm_files_1 folder.
- Browse to core_top.ncf file located in the rpm_file_1 folder with Windows Explorer and comment out all the constraint lines containing U_SET with # and save the file.

```
# Start of RPM Constraints extracted by Floorplanner from the Design
INST "ix147" RLOC = "X0Y0" ;
#INST "ix147" U_SET = "core_top" ;
INST "ix146" RLOC = "X0Y0" ;
#INST "ix146" U_SET = "core_top" ;
INST "core_mid1/shift_core2/reg_qout" RLOC = "X0Y3" ;
#INST "core_mid1/shift_core2/reg_qout" U_SET = "core_top" ;
INST "core_mid1/shift_core2/reg_din_int" RLOC = "X0Y3" ;
#INST "core_mid1/shift_core2/reg_din_int" U_SET = "core_top" ;
INST "core_mid1/shift_core1/reg_din_int" RLOC = "X0Y2" ;
#INST "core_mid1/shift_core1/reg_din_int" U_SET = "core_top" ;
INST "core_mid1/shift_core1/reg_qout" RLOC = "X0Y2" ;
#INST "core_mid1/shift_core1/reg_qout" U_SET = "core_top" ;
```

```
INST "core_mid2/shift_core2/reg_qout" RLOC = "X0Y1" ;
#INST "core_mid2/shift_core2/reg_qout" U_SET = "core_top" ;
INST "core_mid2/shift_core2/reg_din_int" RLOC = "X0Y1" ;
#INST "core_mid2/shift_core2/reg_din_int" U_SET = "core_top" ;
INST "core_mid2/shift_core1/reg_din_int" RLOC = "X0Y0" ;
#INST "core_mid2/shift_core1/reg_din_int" U_SET = "core_top" ;
INST "core_mid2/shift_core1/reg_qout" RLOC = "X0Y0" ;
#INST "core_mid2/shift_core1/reg_qout" U_SET = "core_top" ;
```

- Back in the project navigator window, double click on *MAP* in the Process Window to implement design.
- Open *Map* report by double clicking on *Map Report* under *Map* and browse to section 7.

```
Section 7 - RPMs
----------------
u1/hset
u1/core_mid2/shift_core2/hset
u1/core_mid2/shift_core1/hset
u1/core_mid1/shift_core2/hset
u1/core_mid1/shift_core1/hset
u0/hset
u0/core_mid2/shift_core2/hset
u0/core_mid2/shift_core1/hset
u0/core_mid1/shift_core2/hset
u0/core_mid1/shift_core1/hset
```

- As the map report indicated, Map has generated H_SET from the design. Use this section as a reference, modify core_top.ncf to normalize the each H_SET in order to keep the RPM shape.
  - Cut and paste the following four lines into your ncf.

```
INST core_mid1/shift_core1 RLOC = X0Y0;
INST core_mid1/shift_core2 RLOC = X0Y0;
INST core_mid2/shift_core1 RLOC = X0Y0;
INST core_mid2/shift_core2 RLOC = X0Y0;
```

- Right click on the map report, select re-run all, and check section 7 of the map report.

```
Section 7 - RPMs
----------------
u1/core_mid1/hset
u1/core_mid2/hset
u1/hset
u0/core_mid1/hset
u0/core_mid2/hset
u0/hset
```

- Again, add to the modified NCF file the following constraints to normalize the design. A completed modified NCF file is included in the rpm_files_1 folder. (core_top_mod.ncf). This step is required because we need to normalize the RPM at each level of hierarchy.

```
INST core_mid1 RLOC = X0Y0;
INST core_mid2 RLOC = X0Y0;
```

- Save the NCF file, right click on *Implement Design* in the Process Window and Rerun.
- Check the map report to make sure now only the top level instantiation has been treated as default H_SET.

```
Section 7 - RPMs
----------------
u0/hset
u1/hset
```

- Double click on *View/Edit Placed Design (Floorplanner)* to verify the correct design implementation (Figure 8).

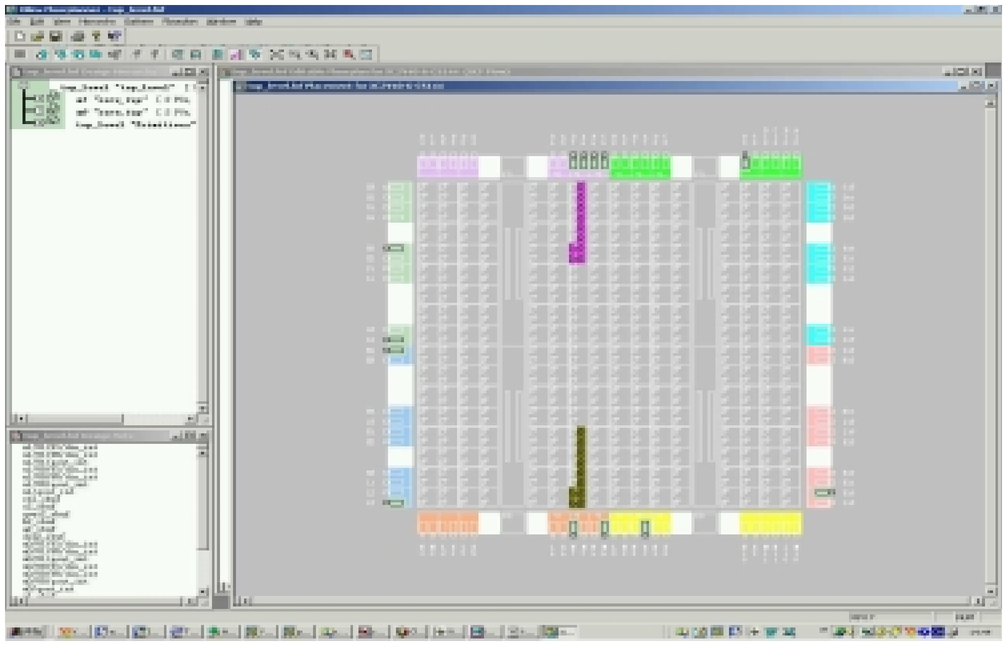

Figure 8: **Implemented View of Workaround D**

- Remember, adding RLOC=X0Y0 at each intermediate hierarchical node is needed at each level of instantiation.

Lab Two Conclusion

Workaround A, uniquification, allows each RPM instantiation to be treated individually. It works for RPMs with or without hierarchical structure. However, since it involves duplicating netlist and ncf files, the project size and file management could be a concern.

Workaround B, flattened RPMs, also allow each RPM instantiation to be treated individually. Since it does not require the duplication of any file, the project size and number of files to manage could be much smaller. However, the RPMs are not always easily captured as flat designs.

Although Workaround C would work on RPMs with or without hierarchical structure and doesn't require duplication of any files, it binds multiple instantiations of RPMs as one RPM. This approach can be cumbersome and inflexible, especially as the number of instances increases.

Workaround D is, generally, the most flexible solution . It works best on RPMs with hierarchical structure and does not require duplication of any files. It keeps each RPM instantiation as separate entity. One minor side case does not work with this method: if the hierarchical node contains only one element in which case MAP will not generate H_SET containing only one element.

This method will be incorporated to a future version of ISE so manual workarounds are not necessary.

Each workaround has its pros and cons. Use the appropriate workaround to better suit each individual design.

Conclusion

After reading this application note and running through the labs, you should have a good understanding of how to use the Floorplanner's "MacroBuilder" capability to generate and use RPMs, work arounds for designs containing multiple RPMs, and the limitation of using XST for the macro builder flow.

For RPMs containing block RAM or multipliers, the use of RPM_GRID should be used to keep the relative placement between sliced logic and block RAM or multipliers. Xilinx Application Note **XAPP416** has more detailed description on implementation of RPM_GRID.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/02/02	1.0	Initial Xilinx release.