# Using the Version 2.1i Xilinx Design Manager and Flow Engine (DMFE)

XAPP403 (Version 1.0) September 27, 1999

Application Note by: Kevin Bixler

## Summary/ Introduction

Welcome to the version 2.1i Xilinx Design Manager (DM) and Flow Engine (FE). The functionality of both DM and FE has been significantly enhanced in this release. In 2.1i, the focus for DM/FE has been to improve "ease of use". A number of new features are provided including "self contained revisions" and the "Smart" Flow Engine, to name a few. These and many other new features are explained in the sections that follow.

## Creating a New Project

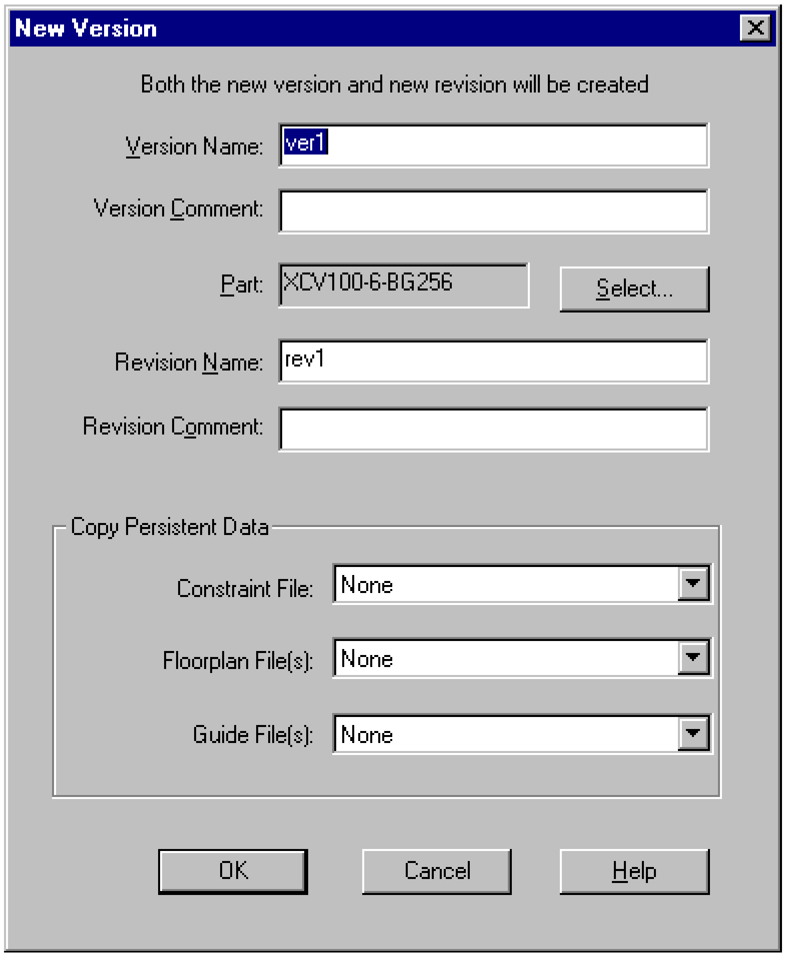
At first glance the 2.1i Xilinx Design Manager and Flow Engine appear to be virtually unchanged from the previous software release. A closer look will reveal the improvements.

When implementing a new design and creating a new project the Design Manager will prompt the user to create a Version and Revision with the *New Version* dialog box, Table 1 and Figure 1 next page.

**Table 1: New Version Dialog Box**

| Section | Description |
|---|---|
| Version | **Version Name** field containing **ver1** and the **Version Comment** field. |
| Part | Non-editable **Part** field and the **Select…** button which spawns the *Part Selector* dialog box. The Design Manager searches the input netlist for a Xilinx part number automatically. If a valid Xilinx part number is found it will be displayed in the **Part** field. The user can override the netlist part number by pressing the **Select…** button and using the *Part Selector*. |
| Revision | **Revision Name** field containing **rev1** and the **Revision Comment** field. |
| Copy Persistent Data | **Constraint File** list, the **Floorplan File(s)** list and the **Guide File(s)** list. The operation of these three settings will be discussed later in this document. |

**Notes**:
1. The 2.1i Design Manager requires at least one design Version with at least one implementation Revision be contained in the Version. These Version and Revision directories are necessary in order to have a location to store the files used in creating the implementation results.
2. When a new project is being created and the *New Version* dialog box is displayed, do not press the Cancel button. Pressing the Cancel button will prohibit the user from proceeding with the project implementation until a new Version and Revision are manually created.


New Version

Both the new version and new revision will be created

Version Name: ver1

Version Comment:

Part: XCV100-6-BG256

Revision Name: rev1

Revision Comment:

Copy Persistent Data

Constraint File: None

Floorplan File(s): None

Guide File(s): None

**Figure 1: New Version Dialogue Box**

.

## Implementing the Design

Once the Project is created containing at least one Version and Revision, the design can be implemented.

The function of implementation has changed in 2.1i. In previous releases of the Xilinx Design Manager, pressing the **Implement** button brought up a dialog box that would allow access to the implementation options.

In the x2.1i release, pressing **Implement** will start the "Smart" Flow Engine running unless the design netlist has changed. If the Design Manager detects the design netlist is newer than the files in the "Last Revision", then the *New Version* dialog box will appear prompting the user to create a new Version and Revision.

**Note**: The *New Version* dialog box will not appear if the **Overwrite last version on implement command** in the **File > Preferences** is checked. If this preference is checked, then the Design Manager will erase the last Version and create a new Revision (Rev1), then begin the implementation.

The most important issues to understand regarding the new function of **Implement** are:

1. The Design Manager does not automatically create additional Versions and/or Revisions unless the design netlist changes. The Design Manager continues to operate on the existing **Last Revision** until the netlist changes or the user manually creates a new Version and/or Revision.

2. The Implementation, Simulation and Configuration options need to be set in the *Options* dialog box, Figure 2, before the user press the **Implement** button.
3. **Implement** will ONLY operate on the **Last Revision** of the Last Version. To implement a Revision other than the Last Revision do the following:
   - Select the Revision the user wishes to re-implement in the Design Manager's project view.
   - Change the user options as desired.
   - Press the right mouse button.
   - In the pop-up menu select the **Re-implement** item. Flow Engine will begin to re-implement the selected Revision.

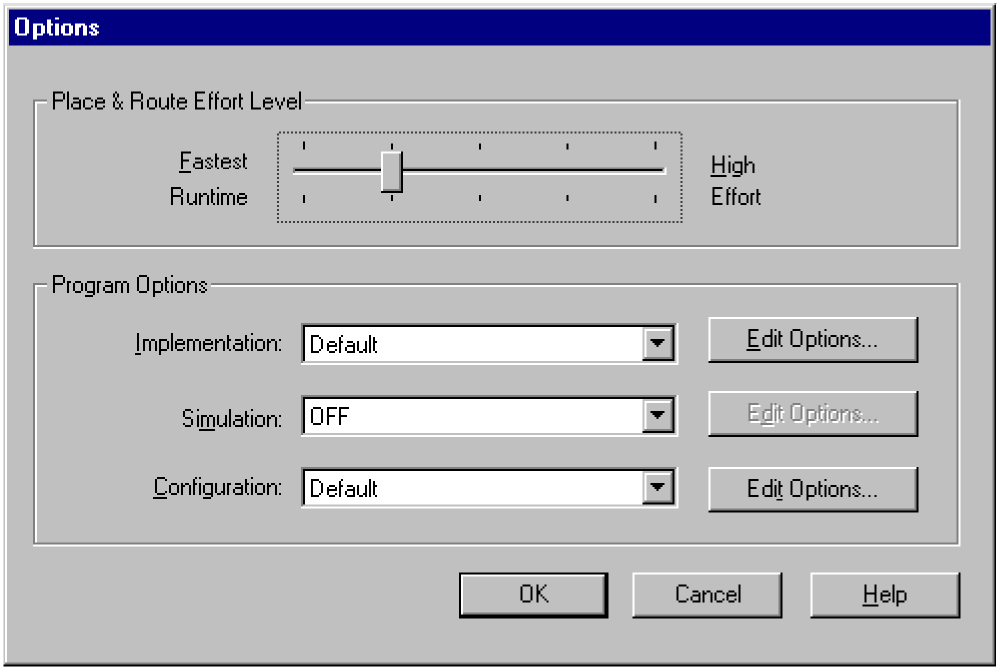
## Implementation, Simulation and Configuration Options

Selecting the **Options** menu item or pressing the **Options** tool bar button will bring up the *Options* dialog box, Figure 2.

A single menu item and the associated tool bar button provide access to the **Implementation**, **Simulation** and **Configuration** options. The **Options** menu item is the second item in the Design Manager's **Design** menu. In the Flow Engine the **Options** menu item is first under the **Setup** menu.

Dominating the new *Options* dialog box is the **Place & Route Effort Level** slide bar. In previous releases, the slide bar was located inside the implementation options, making it less convenient to access.

Control over whether the **Timing Simulation** and/or **Configuration** steps are run is accomplished by selecting a valid set of options or **OFF** from the drop-down list for both **Simulation** and/or **Configuration** (i.e., Default). Notice how selecting **OFF** for Simulation disables (gray's out) the **Edit Options** button as shown in Figure 2.



Implementation: Default

Simulation: OFF

Configuration: Default

**Figure 2: Options Panel**

## Self-Contained Revisions

Now that a Version and Revision has been created and options are specified, implementation can be started.

In 2.1i the implementation, simulation and configuration options are all stored in the individual Revisions. Consequently there must be at least one Version with at least one Revision in order for the Design Manager to have a location for data storage.

**Note**: The last Revision in a Version cannot be deleted without deleting the Version. The last Version in a project cannot be deleted without deleting the project.

In addition to the options files, the constraints file, floorplan files, guide files and almost all other files produced by the Flow Engine will exist in the Revision. The only files that will not be placed in the Revision are the design netlist file, the netlist constraints file (`.ncf`), the `.ngo` file created in the **Translate** step of the Flow Engine and the `time_sim` simulation netlist file created in the **Timing (Sim)** step of the Flow Engine.

**Note**: A method does exist to maintain source netlist information along with implementation output in a Version or Revision. For more information, please reference http://support.xilinx.com/techdocs/6704.htm.

An advantage of having all the files in the Revision directory is that it provides a history of how the results were created.

**Note**: The Clipboard used in previous releases of the software is no longer necessary and has been removed.

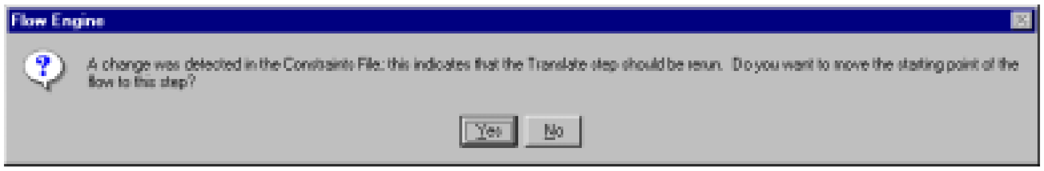
## Making Changes With the 'Smart' Flow Engine

The "Smart" Flow Engine allows the user to make changes to options setting and files such as the constraints file or the floorplan files then have the Flow Engine detect the changes and run the appropriate steps to use the new settings or files. This frees the users from having to understand the details of the steps affected by each option or file. Changes to files are determined by checking the file's date stamp.

As an example of the operation of the "Smart" Flow Engine, consider a design that has completed the **Place & Route** step. For example to change a timing constraint, launch the **Constraints Editor** and make the change, then save and exit. By pressing the **Implement** button in the Design Manager, the "Smart" Flow Engine is launched. It checks the option settings and the files in the Revision. The "Smart" Flow Engine detects that the `.ucf` has changed and issues the message box, Figure 3, prompting the user with the action to take in order for the new `.ucf` to be utilized. The message box shows two buttons: **Yes** and **No**.

Choosing **Yes** will cause implementation to begin, starting with the step that the "Smart" Flow recommends.

Choosing **No** will tell the "Smart" Flow Engine to continue to implement the design starting with the next step that has not completed in the Flow Engine. If a **Stop After** point had been set previously and the design has implemented to the stop sign, the **Stop After** point will be cleared and the implementation will proceed. If the design has previously completed all displayed implementation steps the "Smart" Flow Engine will tell the user that the design has been successfully implemented already, then ask if the user wishes to bring up the Interactive Flow Engine to allow more manual control.



A change was detected in the Constraints File: this indicates that the Translate step should be rerun. Do you want to move the starting point of the flow to this step?

**Figure 3: Flow Engine**

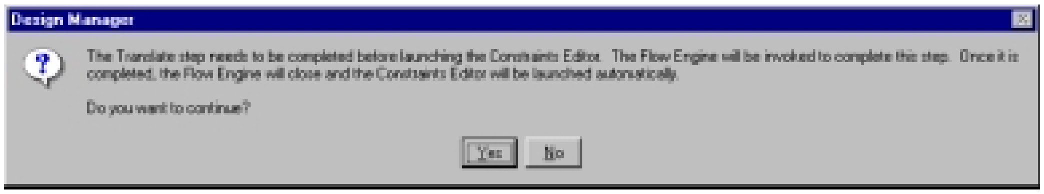
**Note**: If the user decides not to run the step suggested by the "Smart" Flow Engine and presses the No button, the user run the risk of having files in the Revision that do not match those used to implement that Revision. On subsequent (re) implementations using this Revision, "Smart" Flow Engine will again detect the newer .ucf and provide the message asking to rerun the appropriate step.

Another feature of the "Smart" Flow Engine is launching the **Constraints Editor** on a new Revision. The Constraints Editor requires, as input, a `.ngd` file, created in the **Translate** step. If the user select a new Revision when the Design Manager Status shows **(New, OK)**, and selects the Constraints Editor, the message box, Figure 4, will appear explaining that the Flow Engine will run the **Translate** step then exit and invoke the **Constraints Editor**.

The "Smart" Flow Engine will always be used when **Implement** or **Re-Implement** is selected.

If using the interactive Flow Engine (launching the Flow Engine via **Tools > Flow Engine** or the Toolbox button) select **Setup > Update** Flow in the Flow Engine for the "Smart" Flow Engine to detect any changes in the options or files. This was done to allow advanced users that already understand the details of the Xilinx tools flow to proceed without having to respond to various messages from the "Smart" Flow Engine.

**Note**: The "Smart" Flow Engine does not look for changes in the netlist constraints file (`.ncf`). Changes made to the `.ncf` will only be processed if the design netlist changes or if the `.ngo` file is deleted. In either case the Design Manager will prompt the user to create a new Version and Revision for implementation.



The Translate step needs to be completed before launching the Constraints Editor. The Flow Engine will be invoked to complete this step. Once it is completed, the Flow Engine will close and the Constraints Editor will be launched automatically.

Do you want to continue?

**Figure 4: Design Manager**

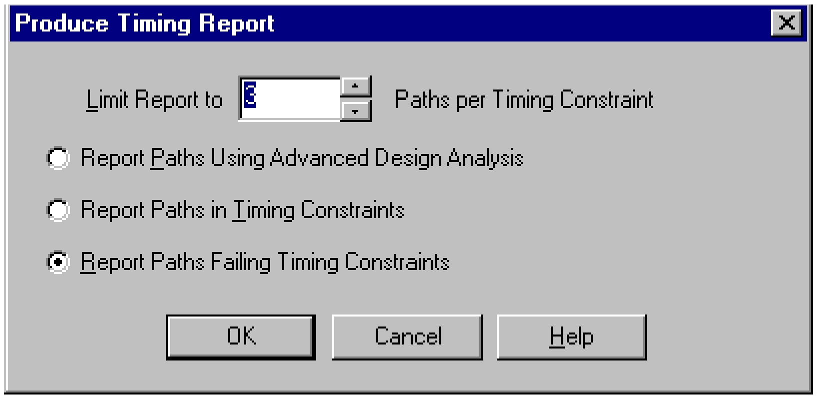
## Produce Timing Report Utility

By default the Flow Engine will produce a **Post Layout Timing Report** using **Report Paths Failing Timing Constraints** to list the three longest paths for each constraint.

The **Produce Timing Report** Utility available in both the Design Manager and Flow Engine allows the user to create various timing reports without having to rerun the **Map** or **Place & Route** steps in the Flow Engine. This will prove valuable in the case of a design that does not contain timing constraints. A timing report using the **Report Paths Using Advanced Design Analysis** will provide the most useful information. This report can be created quickly and efficiently via this utility.

The **Produce Timing Report** Utility allows the user to create the different types of timing reports without invoking the Flow Engine. Selecting this utility will bring up the *Produce Timing Report* dialog box, Figure 5. The name of the timing report created is based on the implementation state and whether reports were previously generated. For example, if a design state is **(Routed, OK)** and the user selects **Utilities > Produce Timing Report** in the Design Manager for the first time, the name of the report generated will be **Post Layout Timing Report #1**. Subsequent reports will be **Post Layout Timing Report #2**, **…#3**, etc. The default setting for producing timing reports is **Report Paths Failing Timing Constraints**.

Additional path tracing, viewing and filtering capabilities are provided in the **Timing Analyzer**.



Produce Timing Report

Limit Report to 3 Paths per Timing Constraint

( ) Report Paths Using Advanced Design Analysis

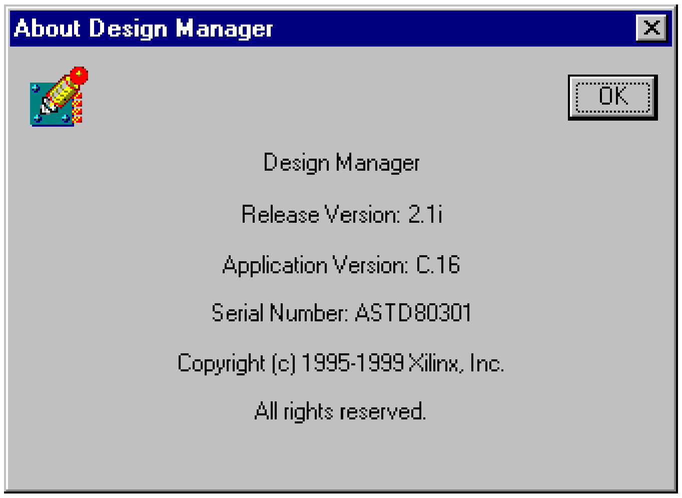
( ) Report Paths in Timing Constraints

(●) Report Paths Failing Timing Constraints

**Figure 5: Produce Timing Report**

## Software Version Information

Using the Design Manager's and Flow Engine's as well as all other Xilinx GUIs, **Help > About** menu item will bring up the *About* dialog box which now provides more detailed information about the software version. This will help users verify which updates have been installed on their systems. An example of the *About Design Manager* dialog box is shown in Figure 6.



About Design Manager

Design Manager

Release Version: 2.1i

Application Version: C.16

Serial Number: ASTD80301

Copyright (c) 1995-1999 Xilinx, Inc.

All rights reserved.

**Figure 6: About Design Manager**

## The 'Copy Persistent Data' Section

The **Copy Persistent Data** section of the *New Version*, *New Revision* and *Set Part* dialog boxes provide flexible and powerful source control. Each of the three controls in this section will be discussed in detail.

- The **Constraint File** setting is used to tell the Design Manager to, copy this user constraints file (`.ucf`) into the new Revision. The `.ucf` will be copied into the new revision as `<design name>.ucf` where `<design name>` is the name of the input netlist. The Design Manager can **only** use `<design name>.ucf` for implementation in the Revision. **The Constraint File** will automatically be set to **None** when creating the first Version and Revision in a Project.

  Normally, when the **Constraint File** is set to **None,** no `.ucf` will be copied into the Revision directory. If no `.ucf` is copied into the Revision the Design Manager will create an empty `<design name>.ucf` in the Revision. The empty `.ucf` is available in the event the Constraints Editor is called. This insures changes made to constraints are stored in the Revision.

The only exception to the normal behavior of the **Constraints File** setting **None** is during the creation of the first Version and Revision in a project. In this special case, if `<design name>.ucf` exists in the same directory as the design netlist, then it will be copied into the Revision directory.

For all following new Versions and Revisions created, the **Constraint File** setting will be **Last Revision (ver# > rev#)** where the # is the last Version and Revision implemented in the project. This setting is used to tell Design Manager to copy forward the `<design name>.ucf` from the last implemented Revision into this new Revision. The user may select from the list of **ver# > rev#** in the project, or use **Custom**.

- The **Custom** setting is used to select a `.ucf` that is either not in one of the project's Revisions or is named something other than `<design name>.ucf`. In the case where the `.ucf` has a different name, the selected `.ucf` will be copied into the new Revision and (re)named `<design name>.ucf`.

**Note**: Whenever a new Version or Revision is to be created, except for the very first Version and Revision in the project, the Constraint File will be initially set to Last Revision (ver# > rev#). For example, if the user had set the Constraint File to point to some .ucf using the Custom setting when the last Version and Revision was created, that Custom setting will NOT be retained!

- The **Floorplan File(s)** setting is used to tell the Design Manager, copy these floorplan files (`.fnf` and `.mfp`) into the new Revision. For the first Version and Revision created in the project, the initial setting is **None**. Unlike the Constraint File setting, **None** ALWAYS means do not copy floorplan files. The Floorplanner does not require Design Manager to create empty files in the Revision when the **None** setting is used. Therefore, empty `.fnf` or `.mfp` files are never created.

  Floorplan files copied into the Revision will always be (re)named `<design name>.fnf` and `.mfp`. For all Versions and Revisions following the very first, the Floorplan File(s) setting will initially be **Last Revision(ver# > rev#)**. This setting provides copying forward of a floorplan file if one existed in the **Last Revision**.

  The Floorplan File(s) setting can be set to **Custom** or any of the project **ver# > rev#** listed. The project **ver# > rev#** list will only contain Versions and Revisions that used the same family, part and package as the part shown in the new Version dialog box. Revisions differing by only the speed grade of the part will show up in the project **ver# > rev#** list.

- The **Guide File(s)** setting is used to tell the Design Manager to "copy this guide file into the new Revision as `guide.ncd` to use as a PAR guide file". The behavior of the **Guild File(s)** setting differs from that of the **Constraint File** and **Floorplan File(s)** settings. **The Guide File(s)** setting remains set to **None** until the user changes it. The **Guide File(s)** setting will retain the user's setting on subsequent Revisions until either the user changes it or the part number is changed. Changing part numbers in any way other than just speed grade will cause the **Guide File(s)** setting to be reset to **None**.

  The behavior of the **Guide File(s)** setting allows the user to retain a pointer to files such as CORE implementations.

- Using the **Custom** setting in the **Guide File(s)** allows the user to set both a **PAR** guide file and a **Map** guide file.

**Note**: Guided Map is not supported for Virtex. If selected, the Map guide file is copied into the Revision as mapguide.ncd.

## The 'Set' Menu Items

Under the Design Manager's **Design** menu, there are five new menu items: **Options, Set Part**, **Set Constraints File**, **Set Floorplan File(s)** and **Set Guide File(s)**. We have already discussed the use of the **Options** menu above. The following covers the other four menu items new to 2.1i.

- **Set Part** allows the user to change the target device they want to use to implement the design. Changing the target device requires a new Revision be created since the available options are tied to the part families. When **Set Part** is selected from the Design menu or the

**Set Part** tool bar button is pressed, the *Set Part* dialog box appears. The *Set Part* dialog box is very similar to the *New Version* dialog box shown in Figure 1 on page 2 except that the **Version Name** and **Version Comments** fields can not be edited.

**Note**: The *New Revision* and *Set Part* dialog boxes are identical except for title.

- **Set Constraints File**, **Set Floorplan File(s)**, and **Set Guide File(s)** all operate on a selected Revision. If no Revision is selected, these menu items are disabled.
- **Set Constraints File** allows the user to copy a user constraints file (`.ucf`) into the Revision AFTER the Revision has been created. Selecting this menu item will bring up the *Set Constraints File* dialog box, Figure 7.

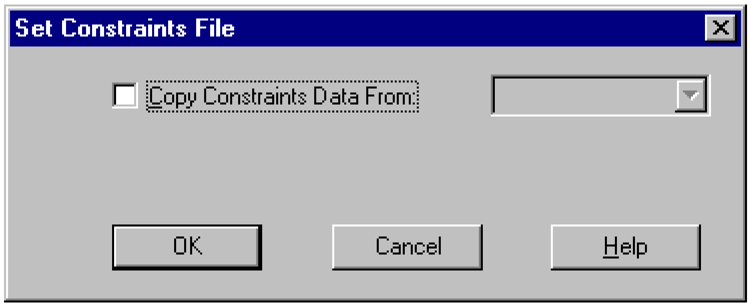


**Figure 7: Set Constraints File**

Checking the box next to **Copy Constraints Data From** will allow the user to select where Design Manager will get the original `.ucf` file to copy into the selected Revision. Remember that no matter what the name of the original `.ucf`, it will be copied into the selected Revision as `<design name>.ucf`.

Set **Floorplan File(s)** allows the user to copy the floorplan files (`.mfp` and `.fnf`) into the Revision AFTER the Revision has been created. It also allows the user to decide if the floorplan file will used in the Revision to implement the design. Selecting this menu item will bring up the *Set Floorplan File(s)* dialog box, Figure 8.

Checking the box next to **Copy Floorplan Data From** will allow the user to select where Design Manager will get the original floorplan files. Remember that no matter what the name of the original floorplan files, it will be copied into the selected Revision as `<design name>.fnf` and `.mfp`. The **Enable Floorplan** switch allows the user to not use the floorplan files during implementation. This switch is on (checked) by default so if `<design name>.mfp` exists in the Revision it will be used during implementation.

**Set Guide File(s)** allows the user to copy a routed `.ncd` file into the Revision as guide `.ncd` AFTER the Revision has been created. It allows the user to decide if the guide file will be used in the Revision to implement the design. This is also where we may decide whether the guide mode is "exact" or "leveraged". Selecting this menu item will bring up the *Set Guide File(s)* dialog box, Figure 9.

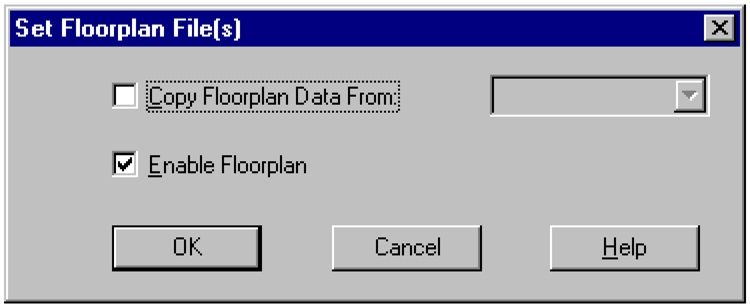
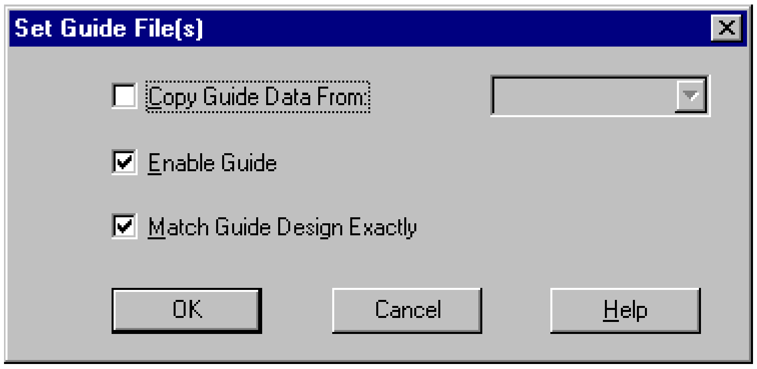


**Figure 8: Set Floorplan File(s)**



☐ Copy Guide Data From:
☑ Enable Guide
☑ Match Guide Design Exactly

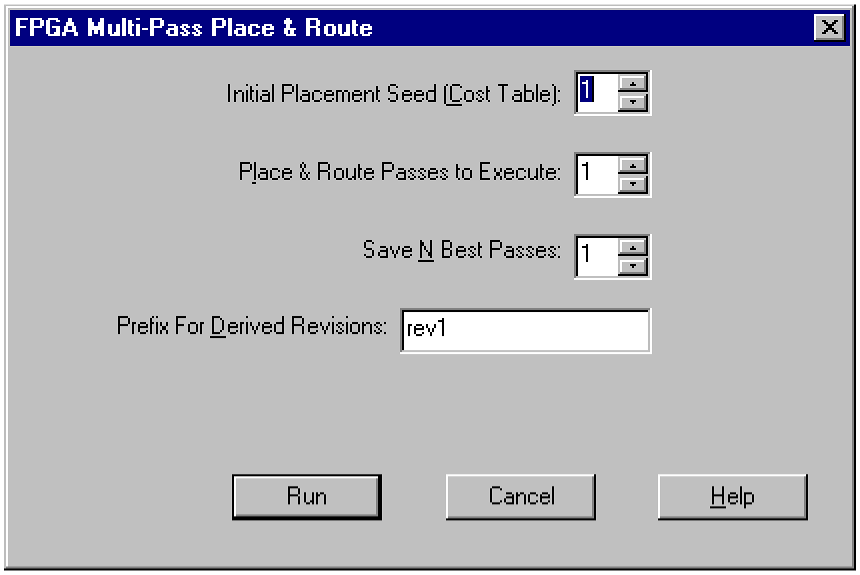
Figure 9: Set Guide File(s)

Checking the box next to **Copy Guide Data From** will allow the user to select where Design Manager will get the original routed `.ncd` file to copy into the selected Revision. Remember that no matter what the name of the original routed .ncd files, the copy in the selected Revision will be named `guide.ncd`. The **Enable Guide** switch allows the user to not use the guide file during implementation. This switch is on (checked) by default so if `guide.ncd` exists in the Revision it will be used during implementation. The **Match Guide Design Exactly** switch, on (checked) by default, allows the user to choose between having to match the design "exactly" or using it as a strong suggestion, "leveraged", for placement and routing (For a more detailed explanation of guide refer to the Xilinx Online documentation).

## Running Multi-Pass Place and Route (MPPR)

MPPR has changed substantially in 2.1i. It is no longer necessary to select a Version in order to enable MPPR. In addition, the new MPPR functionality allows the user to select a Revision that will be used as a starting point for the MPPR run. When a Revision is selected and **FPGA Multi-Pass Place & Route** is chosen, the *FPGA Multi-Pass Place & Route* dialog box is displayed, Figure 10. This dialog box has added **the Prefix For Derived Revisions** field in 2.1i to keep track of how the Revision was derived.

**Note**: There is no longer an Options button in this dialog box. Any implementation options the user would like to set for MPPR must be done prior to selecting FPGA Multi-Pass Place & Route in the Design Manager.


FPGA Multi-Pass Place & Route

Initial Placement Seed (Cost Table): 1

Place & Route Passes to Execute: 1

Save N Best Passes: 1

Prefix For Derived Revisions: rev1

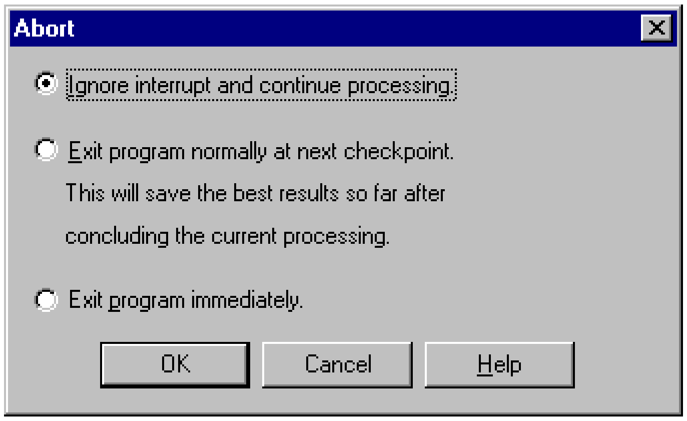
**Figure 10: FPGA Multi-Pass Place & Route**

Also new in 2.1i is the Design Manager's ability to correctly show the implementation status on Revisions created by MPPR even if MPPR is exited early.

When MPPR begins, Design Manager will create a temporary directory in order to store all intermediate results of the process. When MPPR completes, the Design Manager will create separate Revisions for the results based on the setting of **Save N Best Passes**, then delete the temporary directory.

If the user needs to interrupt the MPPR process, the user can press the **Abort** button at the button of the MPPR window. Pressing the **Abort** button while the design is running PAR will bring up the same *Abort* dialog box seen when running non-MPPR based PAR, Figure 11.

One issue with MPPR is the user can not use reimplement on an MPPR created Revision. This is because DMFE will not pass the current cost table to PAR. A work around is to use the **Template Manager** and add the cost table information onto the PAR command line (i.e., -t 3 for cost table three) before the user selects re-implement.


Abort

[✓ Ignore interrupt and continue processing.]

Exit program normally at next checkpoint.
This will save the best results so far after
concluding the current processing.

Exit program immediately.

**Figure 11: Abort Screen**

## Revision History

| Date | Version | Revision |
|---|---|---|
| 09.27.99 | 1.0 | Initial Xilinx release. |
|  |  |  |