APPLICATION NOTE

Mentor Synthesis Group

Managing Projects in Precision

Tom Hill – Synthesis Product Specialist

Mentor Graphics®

Introduction

Performing FPGA synthesis can be an iterative process where constraints and optimization settings are constantly adjusted until the desired result is achieved. Each of these iterations will generate a result based on a specific set of inputs, constraints and synthesis options. Users, performing these iterations often save multiple versions of their design for future reference. Precision Synthesis includes a project management system to help manage this data created during the iterative synthesis process.

Project Management Overview

The Precision usage model is based on projects. The Precision user interface will save all the synthesis input information to a project file which when loaded will restore setup information but will not execute the synthesis process.

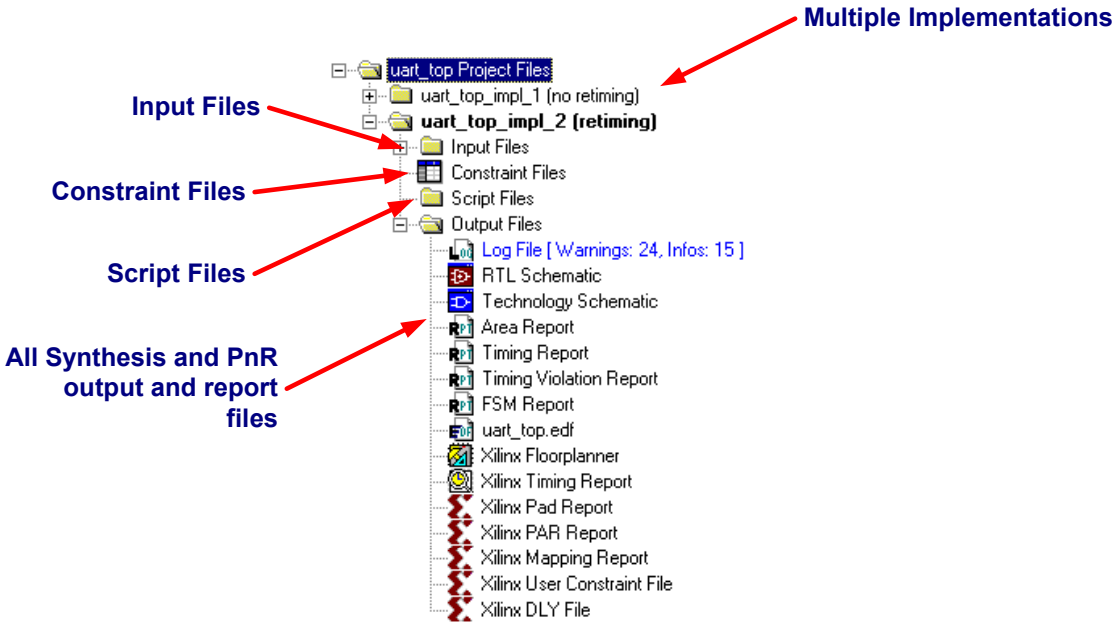

Figure 1 – Precision Project Management File Browser

Precision will save all synthesis and PnR output and report files to the project file.

Creating a Project

The easiest way to save a project is use the Precision GUI to setup a design for synthesis then simply save the project file using the "file -> save project" pulldown menu command as shown below.

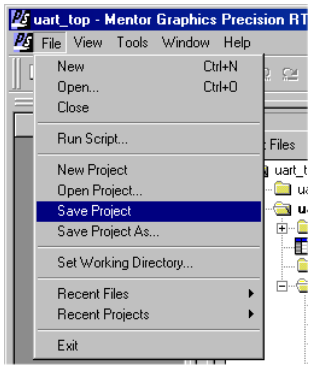

Figure 2 – Saving Projects

A project can be saved at the command line using the "save_project" command

```
> Save_project uart_top.psp
```

Precision, by default, saves the project settings into a file with a ".psp" suffix. This file will include the synthesis setup commands but not the synthesis executing commands.

A project file will contain 5 primary setup commands:

Set_working_dir – Defines the directory or folder where the implementation folder will be placed. This setting will also define the default folder for the GUI when adding input files

Setup_design – Defines all settings that affect optimization including target technology, global frequencies and optimizations settings such as retiming. There are several options to this command

Add_input_file – Will add an input file to the project including RTL, constraints, edif and even non-synthesizable files such as vendor constraint files. Any file added to a project, with an unrecognized format, will be tagged with an "-exclude" property. This will result in the file being copied to the implementation directory by not synthesized

Setup_place_and_route – Setups Precision to run integrated place and route. Precision is tested to specific versions of place and route, which are reflected in the setup_place_and_route command

Setup_analysis – Allows the user to configure the content of the automatically generated reports.

See Appendix A for a project file example

Creating an Implementation

Precision will place the files generated through the synthesis and place and route process into an implementation folder. This provides the benefit of separating the tool-generated files from user-created files such as the RTL source. The implementation folder will be given a default name based on the top-level entity or module name with a "_impl_n" suffix. This name can be overridden using the setup_design command with the "-impl" switch as shown below

```
> setup_design -impl my_implementation_name_1
```

The implementation name can also be set from the GUI using the following project browser pop-up command

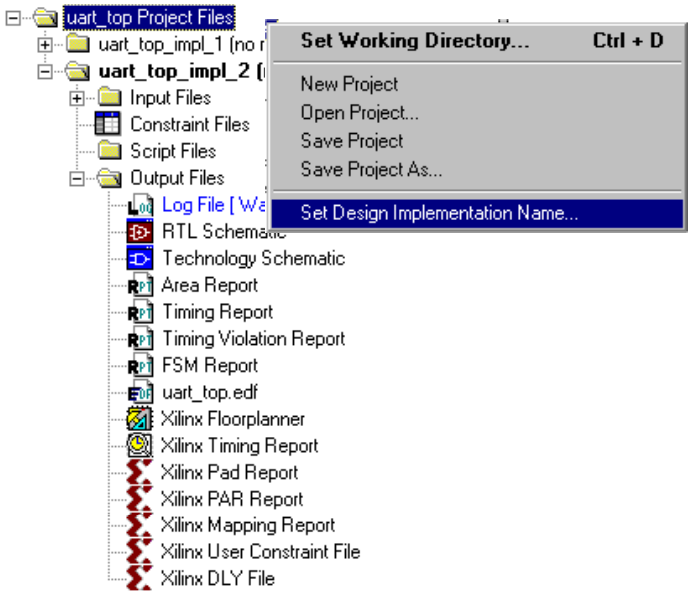

Figure 4 – Setting the project implementation name

Creating a new Implementation Version

Once an implementation name has been set the user can create multiple versions of each implementation. A new version is typically created when performing design iterations. The user can "checkpoint" a particular synthesis results in a revision then experiment with constraint or technology changes. Each revision can have unique technology, constraints and synthesis options. RTL source files are common between revisions.

A new revision can be created from the command line using the following command:

```
> setup_design -increment
```

Or from the user interface using the pop-up command as shown below:

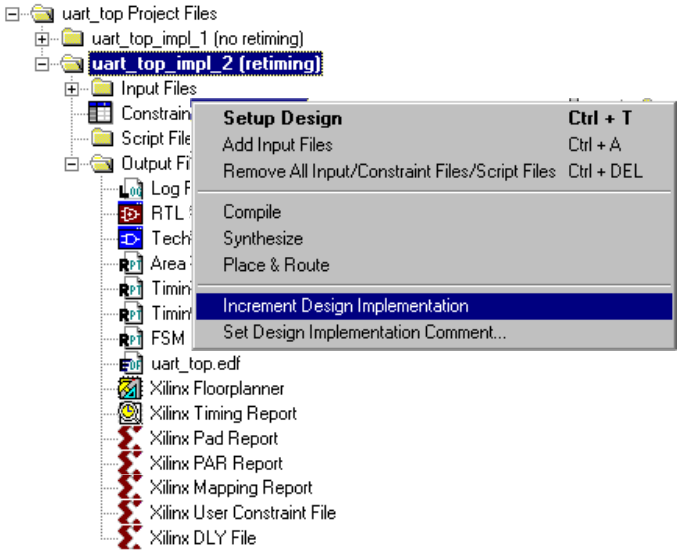

Figure 5 – Incrementing Project Revisions

Switching between revisions

If multiple implementations exist the user can switch between them by simply double-clicking on the version of interest or by selecting a non-active implementation and executing the following pop-up command:

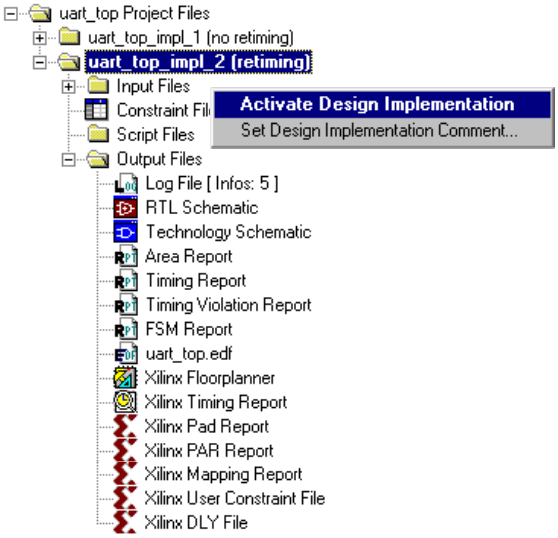

Figure 6 – Switching between Implementations

When a revision is restored the design database is loaded into Precision allowing full design analysis functionality including schematic viewing and incremental timing analysis. Users may even modify constraints and perform incremental timing analysis.

Restoring Projects

Once a project has been created it can be used to fully restore the state of the synthesis tool. Restoring a project will setup Precision with the projects input file list, technology settings and tool settings. If any of the implementation steps have been completed then Precision will also restore all synthesis binary databases and place and route files. Essentially the tool will be restored to its original state when the project was saved allowing users to continue with the synthesis process. The command for restoring a project is "load_project" and is used as follows:

```
> load_project uart_top.psp
```

A project can also be loaded from the user interface using the pulldown menu "file -> open project"

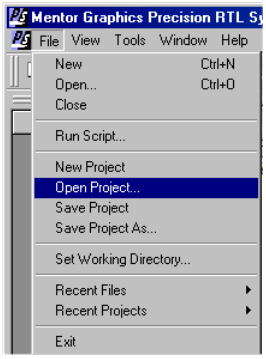

Figure 7 – Open Project

When restoring a project, if multiple implementations exist then the database of the implementation folder with the highest implementation number will automatically load into Precision. The user can override this by using the "*setup_design –impl" <implementation version>" command*. Each implementation has a unique project file that is stored in the implementation directory and contains that implementation's unique settings. The master project file will restore the original project settings that then the implementation project file will automatically load to override any changes unique to that implementation. The master project file includes the settings that exist when the "save_project" command is executed. An implementation's project settings will be automatically saved after each synthesis run. Users are not required to explicitly save this file

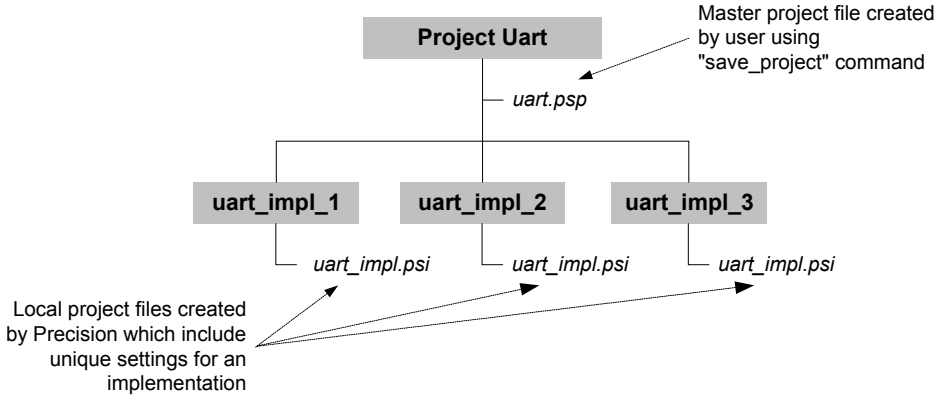

Figure 8 – Hierarchical project file structure

Specifying Output Files

Precision provides users complete control over the naming and location of output files. This is accomplished through the *"setup_design"* and *"set_working_dir"* commands. To specify the location and name of an output file use the following commands:

```
> Set_working_dir c:/designs/uart
> Setup_design –impl uart_imp_1
> Setup_design –basename my_design
```

This will place the output EDIF file into c:/designs/uart/uart_imp_1/my_design.edf

Note: *Precision uses TCL for a scripting language. TCL requires the use of forward slashes "/" to specify directory structures, even on Windows OS.*

Project Files vs. Scripts

As stated earlier the purpose of the Precision project file is to setup Precision for synthesis on a particular design. Once setup Precision includes 3 commands for executing the implementation flow

Command	Function
Compile	Will compile the RTL source into a generic netlist format. Once compile is complete users can set constraints or view RTL schematics
Synthesize	Will perform optimization and generate all netlists, vendor constraint files and reports necessary for place and route
Place_and_route	Will launch vendor specific place and route tools and automatically perform place and route.

Any project file can be converted into a script file by progressively adding these commands.

Users may add constraints directly into a script file. Project files require either a separate constraint file or use of a global frequency setting. If the user wishes to add timing or optimization constraints to the script file they may do so after the compile command and before the synthesis command. A short example is provided below

```
# Working Directory
set_working_dir C:/DAC2002/demos/basic_flow

# Add the input files
add_input_file {traffic.v}

# Technology Settings
setup_design -manufacturer Xilinx -family VIRTEX-II -part 2V40cs144 –speed

# Compile the design
Compile

# Setup timing constraints
Create_clock clk –period 10

# Finish the implementation process
Synthesize
Place_and_route
```

Appendix A – Example Project File Example

```
## Working Directory
set_working_dir C:/DAC2002/demos/basic_flow


## Technology Settings
setup_design -manufacturer Xilinx -family VIRTEX-II -part 2V40cs144 -speed 6


## Input File Settings
setup_design -design=uart_top
setup_design -arch=
add_input_file -format Verilog -work work -compile_time 1047222896 address_decode.v
add_input_file -format Verilog -work work -compile_time 1047222896 uart_top.v
add_input_file -format Verilog -work work -compile_time 1047222896 clock_divider.v
add_input_file -format Verilog -work work -compile_time 1047222896 control_operation.v
add_input_file -format Verilog -work work -compile_time 1047222896 cpu_interface.v
add_input_file -format Verilog -work work -compile_time 1047222896 mit_rcv_control.v
add_input_file -format Verilog -work work -compile_time 1047222896 serial_interface.v
add_input_file -format Verilog -work work -compile_time 1047222896 status_registers.v


## Output File Settings
setup_design -basename=uart_top


## Design Settings
setup_design -addio
setup_design -vhdl=false
setup_design -verilog=false
setup_design -edif
setup_design -vendor_constraint_file
setup_design -transformations
setup_design -retiming=false
setup_design -advanced_fsm_optimization
setup_design -use_safe_fsm=false
setup_design -encoding=auto
setup_design -resource_sharing
setup_design -fault_tolerant=false
setup_design -frequency=100
setup_design -radhardmethod=
setup_design -input_delay=0
setup_design -output_delay=0
setup_design -search_path=

## Design flow and state information
go -set_flow rtl -set_state place_and_route

## Active Implementation
setup_design -impl uart_top_impl_2
```