Method to Instantiate and Use a Core in Synplify™

Introduction

This application note is intended to assist people who use cores for Cypress CPLDs and compile their design in *Synplify*™. These cores are distributed using the VIF file format which is generated by *Warp*™. This note contains a detailed description on how to use cores and associated wrappers in *Synplify*. Some cores may be parametrized using VHDL generics or Verilog parameters. Instructions on how to use both parametrized and non-parametrized cores are described in this document. Cores and associated wrappers can be downloaded from the IP vendor's web site. For more information on how to create a wrapper, please see application note "Method to Instantiate and Use a Core in *Warp* with Cypress CPLDs".

Using VIF Files Without Generics in *Synplify*

This section is designed to show the user how to use the downloaded VIF files and associated wrapper in *Synplify*. The core described in this section is not parametrized.

1. In the wrapper of the VIF files add the following lines at the beginning of the wrapper before the entity section.

   library Synplify;

   use synplify.attributes.all;

This will allow the black box attribute of *Synplify* to be included. By using the black box attribute, the core will pass through *Synplify* smoothly. *Synplify* will recognize the ports of the entity of the core, but will ignore the contents of the core. The black box core is eventually linked and fit by *Warp*.

2. Add the following lines before the 'begin' reserved word in the architecture section.

   attribute syn_black_box: boolean;

   attribute syn_black_box of arch_mi2cfromvif: architecture is true;

The above declares a black box and associates this black box with the architecture of the wrapper. The "arch_mi2cfromvif" is the architecture name of the VIF wrapper.

3. Save the file. The wrapper is now ready to pass through *Synplify* successfully and can be instantiated in the user's own design.
4. Compile the wrapper and the user's design file that contains the wrapper instantiation in *Synplify* and map to Delta39K. In *Figure 1*, the wrapper name is p2swrapper.vhd and the user's design name is p2swrappertop.vhd. The arrow indicates that this Synplify project maps to Delta39K device.

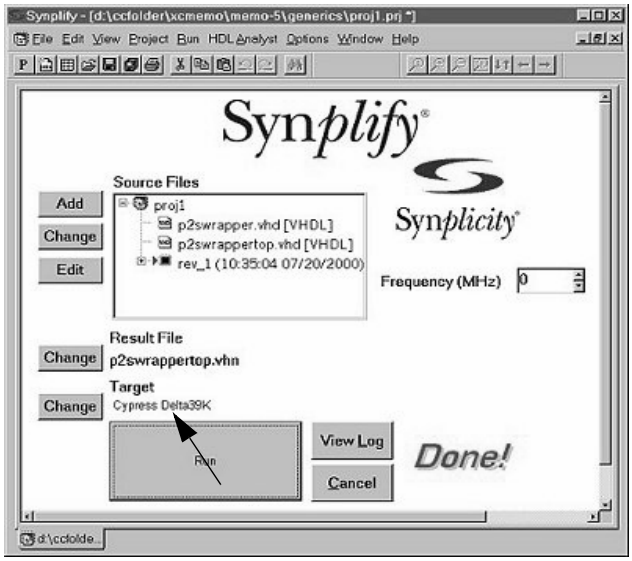

Figure 1. Synplify Project Containing the Wrapper and the User's Design

5. Before the VHN file can be added to the project, go to "Edit -> Preferences..." and type vhn in the area shown by the arrow in *Figure 2*.

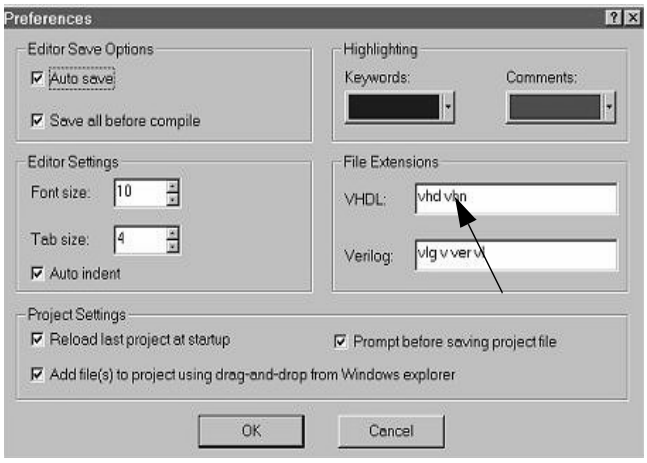

Figure 2. Setting Preferences to Allow VHN Files to Compile in *Warp*

6. Create a new project in *Warp* and bring the resulting VHN files from *Synplify* into *Warp*. The VIF files are placed in a directory that has the same name as the device selected with a 'lc' prefix. In this example, VIFs are fit to a 39K100 device, and placed in a directory called 'lc39k100'. The path of this directory is the same as the path of the project created in this step.
7. Open the VHN file and find the entity declaration of the user's top-level design component. Before the reserved word 'entity', add the following line:

   use work.componentname;

The componentname is the entity name of the wrapper file.

Using VIF Files With Generics in *Synplify*

This section is designed to show the user how to use the downloaded VIF files and its wrapper in *Synplify*. The core described in this section is parametrized, i.e. contains generics.

1. Before making any changes to the downloaded wrapper, make a copy of this wrapper. This copy will be used in step 9 of this section.
2. In the wrapper of the VIF files, add the following lines at the beginning of the wrapper before the entity section

   library Synplify;

   use synplify.attributes.all
3. Before the 'begin' in the architecture section, declare a black box and associate it with the wrapper. Also, identify the generics of the VIF files from the wrapper. For example, if the wrapper's architecture section is called "arch_wrapper" and the generics of this VIF component are 'resettype', 'width' and 'length', the following code will declare the black box and the generics in *Synplify*.

   -- black box declaration
   attribute syn_black_box: boolean;
   attribute syn_black_box of arch_wrapper: architecture is true;
   -- generics identification
   attribute \resettype\: character; -- all generic type is character
   attribute \width\: character; -- needed by synplify.
   attribute \length\: character;

Notice the "\" before and after the generics, which must be present to compile successfully in *Synplify*.

4. Save the wrapper. The wrapper alone can not be passed into *Synplify*. It must be instantiated in another design that does not contain generics in its entity section.
5. Compile the wrapper and the design that contain the wrapper instantiation in *Synplify.*
6. Before the VHN file can be added to the project, make sure the following option is set. Go to "Edit -> Preferences..." and make sure that in the "File Extensions" section, the vhn files are part of the VHD file extension. This is shown by the arrow on *Figure 2*.
7. Create a new project in *Warp* and add the resulting VHN file to the project.
8. Open the VHN file and find the entity declaration of the user's top-level design. Before the reserved word 'entity', add the following line:

   use work.componentname;

The componentname is the entity name of the wrapper.

9. Add the original wrapper file copied in Step 1 to the new project created in step 6. Compile this wrapper by selecting it and going to "Compile -> Selected File(s)". Compile the VHN file from *Synplify* by hitting the '**Compile Project**' button shown below.

This is the '**Compile Project**' button located on the toolbar

10. Go to "Project -> Library Manager -> Assign". If there is nothing in this screen, click on the '**add**' button. Select the work library and press '**OK**'. This is shown in Figure 3. Switch to the libraries folder and open the work library. Make sure the wrapper name is shown in the library. This is to ensure that the core in the VIF file is correctly linked to the project.

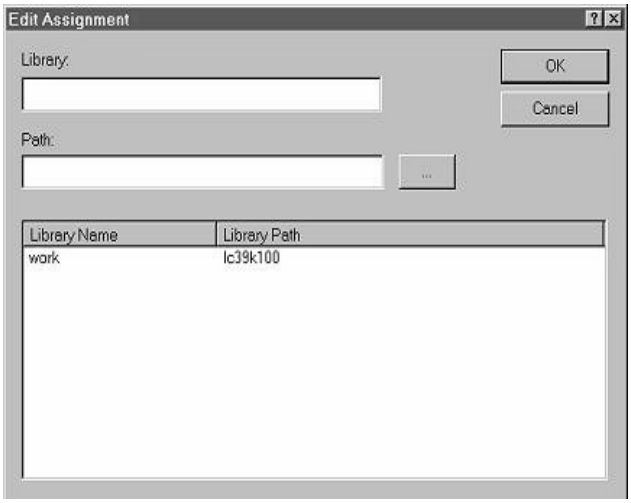

Figure 3. Add a Library to the *Warp* Compiled Library

Conclusion

The steps described in this document, allow the user to use Cypress cores together with *Synplify*. This compatibility between Cypress cores and *Synplify* gives users greater flexibility and support.