Instructions on creating CTL files for PAL pin placement

We use CTL files to control how Active-HDL assigns inputs and outputs to pins. This will make it easier for you to re-synthesize a file since Active-HDL will place the inputs and outputs on the same pins. Keeping the pin assignments identical will allow you to replace the chip with a new program without rewiring your circuit. Remember to pay close attention to how many terms an output requires as the outputs in the PAL do not have the same amount of terms.

To create a CTL file:

CTL files follow a specific format so please adhere to these rules closely.

1) Create a blank file and name it <module_name>.CTL, where <module_name> refers to the name of the module that was set as the Top Level for synthesis. In the case of a verilog file the <module_name> is defined by the keyword module. In the case of a block diagram the module name is the exact same name as the block diagram name (i.e. <module_name>.bde>

   For example, if we wanted to make a wiring file for the module full_adder we would name the file: "full_adder.CTL". You will probably want to use some text editor to create the file. NOTE: Do not let the text editor you are using append any extensions on the end of the file name. Make sure your file does not get named something like "full_adder.CTL.txt".

2) The first line of your CTL file must match the name of the CTL file and the module name:
   `attribute pin_numbers of <module_name>:module`
   where <module_name> refers to the name of the module that was set as the Top Level for synthesis. The rest of the CTL file is determined by where you want to place the inputs and outputs. For example, the first line might resemble `attribute pin_numbers of full_adder:module` for a module named full_adder.

3) Choose the pin number that you want to assign to each variable during synthesis & implementation. Keep in mind the following pin assignment schemes:
   a. **Pin 1: clock/input** (Only pin that can be assigned to a clock)
   b. **Pins 2-11 & 13: inputs**
   c. **Pins 14-23: outputs** (can also be assigned to be inputs; however, outputs are limited so not recommended unless needed)

   In addition, some of the output pins can handle more product terms than others. You need to consider product term sizes when making your CTL or Active may ignore your assignments. If Active ignores your CTL, see "fixing mistakes" section.
   The breakdown of product terms for output pins is:

Pin:	**Product Term Capacity**
14:	8
15:	10
16:	12

17: 14
18: 16
19: 16
20: 14
21: 12
22: 10
23: 8

4) The next step is to assign the pins to the desired variable name. Pins are assigned in a string ending with a semi colon. For readability most people write their pin assignments on separate lines. The instructions focus on a pin assignment per line method but you may write a single string if you wish.

    a. Each pin assignment line should follow this format with the exception of the last line of your CTL file.
    "<variable_name>:<pin number> " &

    example: "A:2 " & (where A=<variable name>, 2 =<pin number>)
    example for bus: "B(0):2 " & (Note: Bus name then the index in parentheses)

    Spacing and exact placement of quotation marks is important as you are creating a string to assign the pins. The space is needed to separate the pin assignments as strings will be joined into a single string.

    b. For the last line of your CTL file you will want to put a semicolon to mark the end of your string. Follow this format:

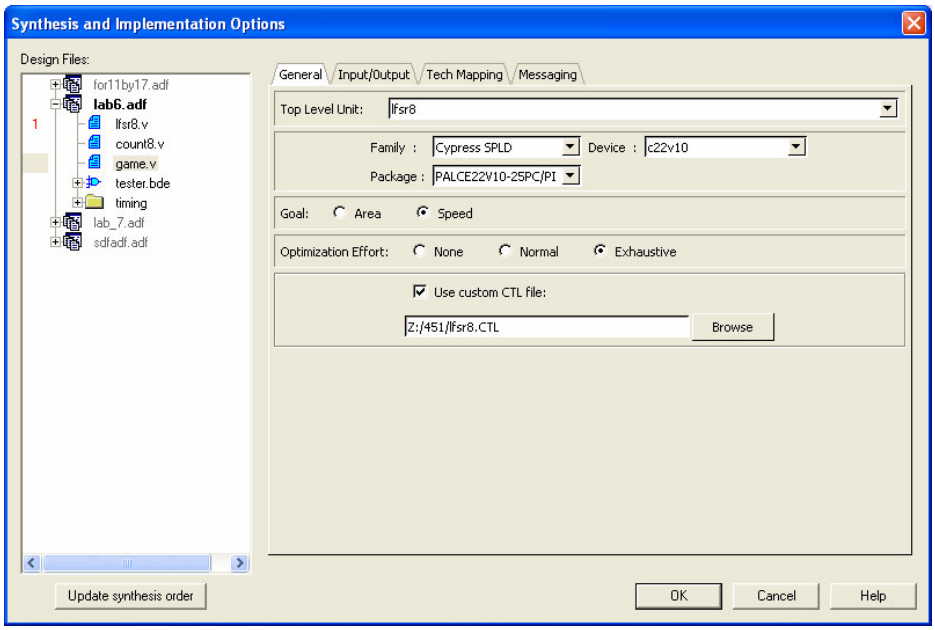
    "<variable name>:<pin number>";

    Example: "Sum:20";

5) Now all you need to do is include the CTL file at synthesis time. Refer to the next section on "Using the CTL file for synthesis & implementation".

Using the CTL file for synthesis & implementation.

1) Click on the options button (located next to the Synthesis & Implementation button) to include your CTL file during synthesis & implementation.
2) Check the box: "Use custom CTL file". Then, browse until you find the path to your custom CTL file. See the figure that follows.



3) Click OK. You are now ready to run synthesis & implementation. If everything goes well the pin assignments in the report file will match the pins assignments in your CTL file. Otherwise, see the "fixing mistakes" section.

Making a CTL file: Full Adder Example

For example, here is the Verilog code for a full adder:

```
module full_adder ( A ,Cin ,B ,Cout ,Sum );

input A ;
wire A ;
input Cin ;
wire Cin ;
input B ;
wire B ;

output Cout ;
wire Cout ;
output Sum ;
wire Sum ;

assign  Sum = (A & ~B & ~Cin) | (~A & B & ~Cin) | (~A & ~B & Cin) | (A & B & Cin);
assign  Cout = (A & Cin) | (B & Cin) | (A & B);

endmodule
```

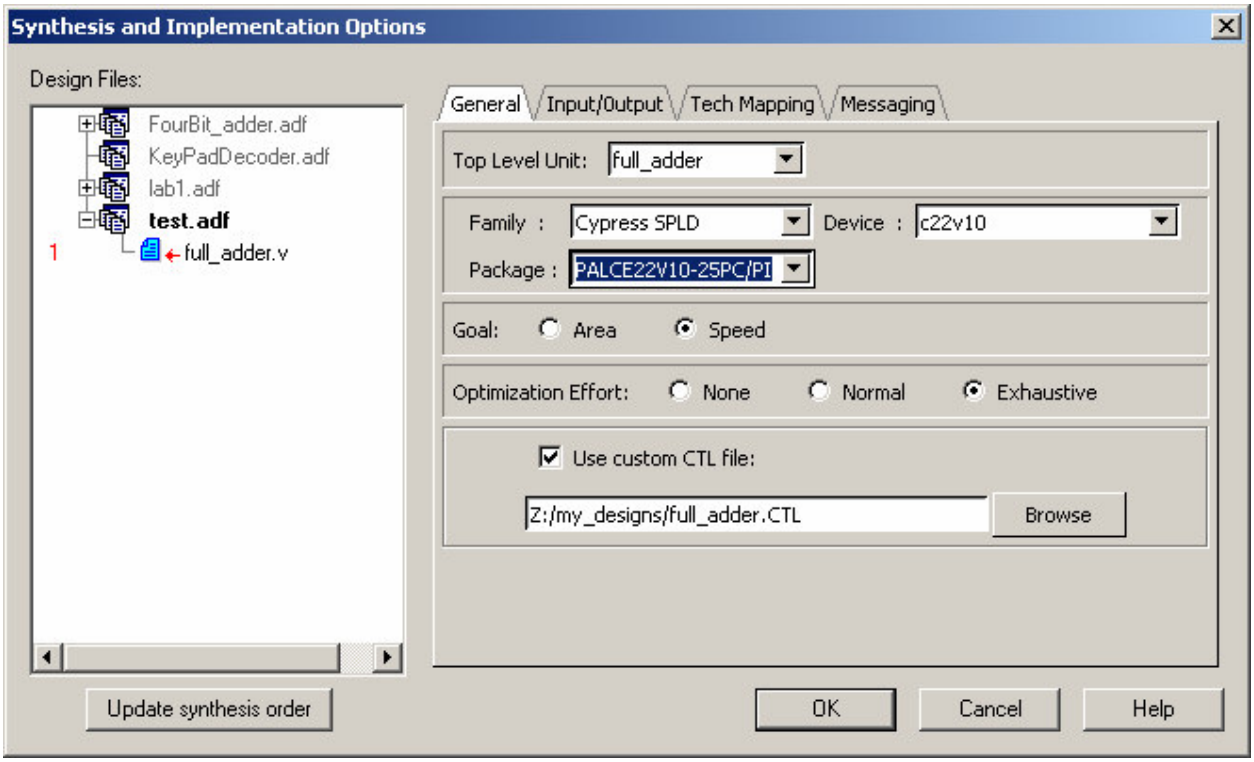
1) From step 1 in the creating a CTL file section we know that the name of the CTL file for this full adder will have to be "full_adder.CTL". Open a new file, and save it as "full_adder.CTL"
2) Looking at the code there are 5 inputs and outputs:

A
B
Cin
Sum
Cout

3) If you wanted to assign A to pin 2, B to pin 4, C to pin 6, Sum to pin 20 and Cout to pin 18, an example CTL file would be:

```
attribute pin_numbers of full_adder:module
"A:2 " &
"B:4 " &
"Cin:6 " &
"Sum:20 " &
"Cout:18";
```

In particular, notice the spacing and the use of ampersands for concating the string and a semicolon to end the string. If you don't have the spaces the pin assignments will run together. These are all critical to the CTL working.



Fixing Mistakes

If you miss one of the steps above, one of following two things will occur:

1) Active-HDL will ignore your CTL file and do whatever it wants with pin assignment. Active-HDL may then ignore subsequent attempts to fix the CTL file. Try the fixes mentioned below.
2) Active-HDL may spit out error messages to you about your CTL (missing a space, cannot understand this word). If you're lucky enough to get these error messages, try to follow them and, if that doesn't work, use the fixes mentioned below.

The fixes:

1) First try to delete the .jed file within the synthesis folder, along with the .PIN file, that has the same name as the file you are trying to synthesize.
2) Now, retry synthesis with the correct CTL file
3) If this works, great. Otherwise, delete the synthesis folder (scary, but there should be no code within the synthesis folder). Anything .jed files that you had previously synthesized will be lost.
4) Now, retry synthesis with the correct CTL file. If your syntax all matches within the file, and the naming conventions are correct, everything should be ok.
5) You will need to re-synthesize all files that you need a .jed to program.