Using the Core Generator Tools
Project: Audio Processing Unit
Submitted by: Clint Lozinsky, Duncan Campbell, Grant Cunningham, Richard Schultz

Introduction

Reusable designs are a key concept in the world of electronic design automation these days. This methodology of design enables companies to resuse, sell and purchase fully tested, complete cores designed for a specific purpose. Many companies today exist solely for the purpose of creating nothing but intellectual property; they sell no tangible goods, only ideas. These intellectual property cores greatly decrease design time, and vastly expand what is capable in today's hardware design world.

The purpose of this document is to give a brief overview on how to use the Xilinx Core Generator tools for ISE 4.2i to implement third-party cores into a design. It is intended to enable the reader to quickly get an understanding of the tools and to be able to generate cores and use them effectively in future projects. This document assumes the reader is familiar with the Xilinx ISE package, and has a decent understanding of VHDL.

Procedure

For starters, a project file is needed. If you plan to include the core you generate into an existing project, this step isn't needed, but for the purposes of this tutorial, I'll assume this hasn't been done.

Once a project has been created, open up the Xilinx Core Generator System
In Windows:
"Start -> Programs -> Xilinx ISE 4 -> Accessories -> Core Generator System"

Next, Select the project you wish to add a core to.

Once you have done this, a window will pop up, asking the user to select certain design parameters. The only change that should be necessary at this point is the target architecture (SpartanII for the Audio Processing Unit).

Following this, the user is given a list of cores in the left hand pane of the window. Depending on the target architecture, there are many different cores available that do everything from basic math functions to complete Microprocessors. For the purposes of this tutorial, we'll implement a basic fast fourier transform (fft).

From the left pane:
"Digital Signal Processing -> Transforms -> FFTs"
Then, double click on **"32-Point Parameterisable Complex Fast Fourier Transform"**

Next, you must assign a name to the core and change the paramters. We'll set the name as fftcore, and to make sure this core will fit we're going to change the precision to 4 bits for both of the options. Depending on the core you need, there are literally hundreds of different customizable options for each core. After selecting the parameters, simply select "Generate Core". After a few seconds, a window should pop up saying that the core was successfully generated.

Following this, the project file in Project Navigator is automatically updated. There should be a file called fftcore.xco in the "Sources in Project" window. All thats needed at this point is to instantiate the core.

The quick way to do this is to use fftcore.vho. This is an instantiation template file that contains all the code necessary. Using the template as a guide, a quick and easy fft.vhd (included in the fft.zip file linked at the end of this document) file can be created that fully instantiates the core that was just generated into a project.

From here, the vhdl code is fully synthesizable in hardware, and can be implemented as any other vhdl entity would be implemented. As you can clearly see, this has cut down drastically on design time, and greatly improves design efficiency.

Files
http://www.ee.ualberta.ca/~elliott/ee552/studentAppNotes/2003_w/cad/core_gen/core.zip

Links
Xilinx's list of Intellectual Property that the Core Generator has access to
http://www.xilinx.com/xlnx/xil_prodcat_landingpage.jsp?title=Intellectual+Property

Free IP – An open source site dedicated to Intellectual Property
http://www.free-ip.com/

Opencores.org – Another open source site dedicated to Intellectual Property
http://www.opencores.org