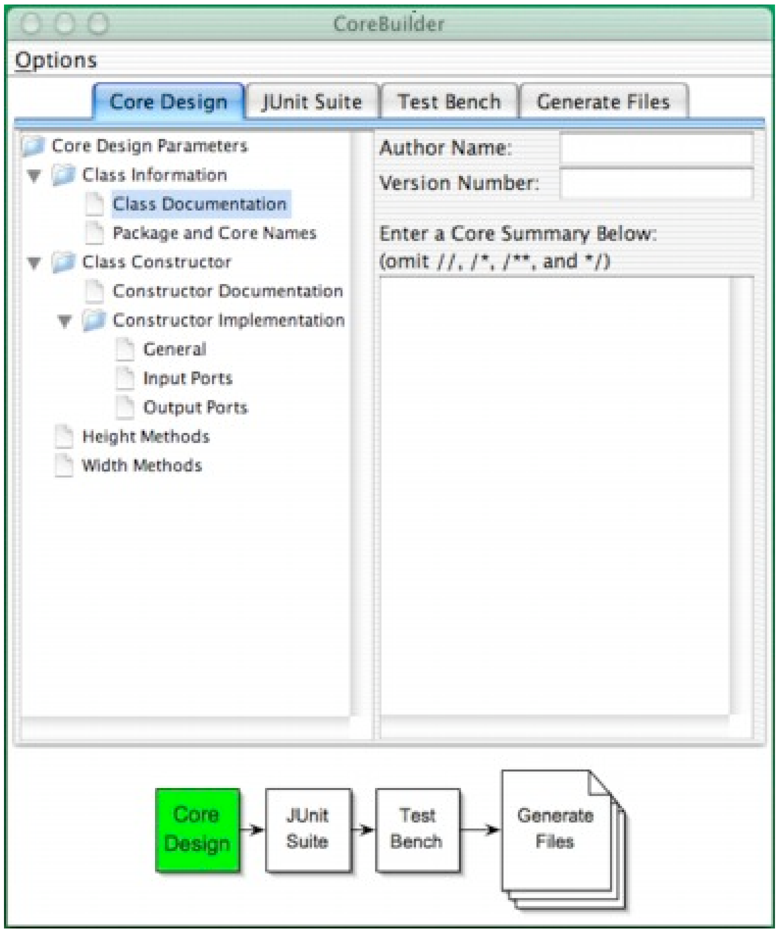
CoreBuilder 1.0 Release Notes

Overview:

CoreBuilder is a simple program that is designed to speed the development and testing of JBits RTP cores.



A scaled screen shot of CoreBuilder 1.0 on Mac OS X

CoreBuilder generates code that is the "frame" of a JBits RTP core, the user is left to fill in the implementation details of the core in the generated code. The most powerful feature of CoreBuilder, is the test bench generator which allows the user to generate a core which provides test vector inputs to another core. CoreBuilder can also generate partial JUnit files for unit testing a core.

History:

CoreBuilder was written while I was completing my master's thesis as a wizard to help me rapidly develop JBits RTP cores. I found I was writing between 5 and 50 cores a month and got sick of cutting and pasting the code that is the outline of a JBits RTP core. Furthermore, I found I needed a quick way to generate a test bench for all the cores I was writing. Knowing little about GUI programming, I set out to write CoreBuilder so that it would support Java Virtual Machine 1.2. The result is rough, but I found it useful enough (particularly for cores with a lot of IO) and decided to release it. I hope you find it useful too.

If you have any comments, requests, or bug reports, email me at aycarrei@shaw.ca.

System Requirements:

1. The Xilinx JBits API is installed. (Preferably 2.8 or later—RTP core support must be present at the very minimum).
2. JUnit is installed (if you plan on using JUnit code generated, test w/ version 3.7).
3. A Java Virtual Machine (>= 1.2).

Installation:

ICoreBuilder 1.0 has been tested in Mac OS X, Windows XP Home, Windows 95, and Windows 2000. The following instructions should get you running in any of these operating systems:

1. Download CoreBuilder.jar to a directory of your choice.
2. Open a console window and navigate to the directory containing CoreBuilder.jar.
3. Try typing "java -jar CoreBuilder.jar".

You should get the following message "Unable to find or read a preferences file.", then CoreBuilder should start up in new frame. If this does not work, ensure that you have a Sun JDK installed and that the file "java.exe" is in your system path. If a valid JDK is installed (>= 1.2) and the file java.exe is in your system path, email me the error message and I will try to help you get up and running.

Also, note that in many systems, placing CoreBuilder.jar on the desktop or in a folder and double clicking on it will start it up as well.

Test Bench How-To:

Generating Test Benches

Test Benches can be generated for existing cores as well as cores that you are generating with CoreBuilder. To generate a TestBench for the existing core with the following particulars:

Core Name:	Add
Input Ports:	a, b, Clock, enable
Output Ports:	c

1. Under the "Core Design" tab, enter the core name and add all of the input and output ports.

2. Under the "Test Bench" tab, for each Input Port:

   a) Select the desired input port in the list to the left.
   b) Choose a connection type.
   c) Enter a test vector if the type is "hex" or "bin".

In 2.c if you were to enter a hexadecimal test vector "AF", a test vector core would be generated as an SRL 16 with the contents "0000 0000 1010 1111". The binary SRL contents would be shifted out to your core from right to left, that is 1 would be the first output followed by three more ones, then a zero and so on.... Because of the way the SRL 16 core is implemented in the test vector cores used in the test bench, eventually the vector will go to a Logic 1 state. Keep this in mind when testing.

3. Under the "Generate Files" tab:

a) Choose a save directory for the test bench by clicking on the "..." button.
b) Click "Generate Test Bench File".

4. You might have to add an import line in the test bench java file for it to compile (import the core that the test bench connects to).

The test bench will input values to the core you have created. You must know where Flip-Flop outputs and other resources are located in your core so that you may decipher your core's response to the test bench inputs. It is often handy to connect these resources to large serial registers which allow you to readback the core's response to the test bench inputs.

Tips:

1. You can kill the splash screen that appears in the startup in Options->Preferences.

2. The choosing output directories dialog (activated by the "..." buttons) may require that you highlight the desired directory and may not work if you are in a directory with no highlighted directories.

License & Disclaimer