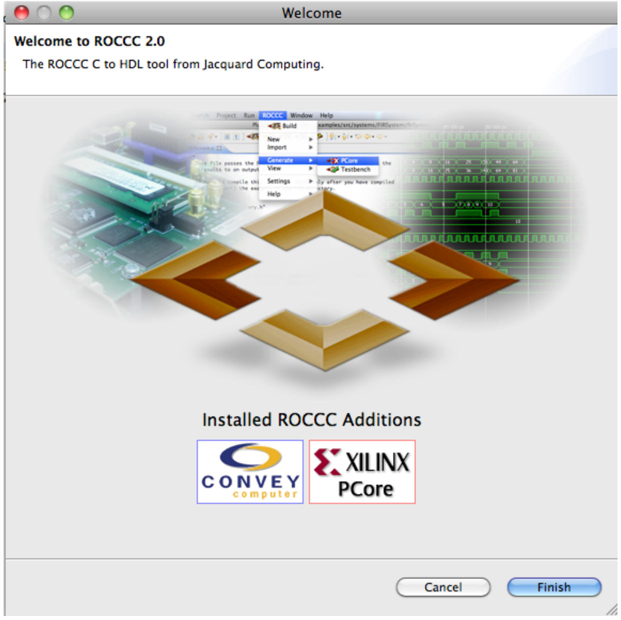
ROCCC 2.0 Convey Port Generation - Revision 0.7.5

January 22, 2013



Welcome to ROCCC 2.0

The ROCCC C to HDL tool from Jacquard Computing.

Installed ROCCC Additions

Figure 1: Splash Screen

1 Usage

1.1 Installation

The Convey HC-1 port requires the latest version of ROCCC. To install the Convey HC-1 port you must copy the plugin into the eclipse/plugin directory. The appropriate features will be loaded up the next time you run ROCCC through eclipse. When the Convey plugin has been installed the opening splash screen will display the Convey logo, as in Figure 1.

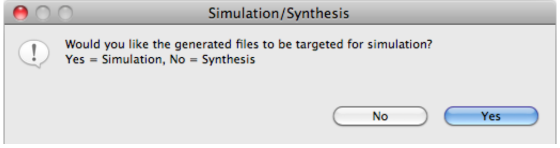
1.2 Interface Generation

The Convey HC-1 port adds the ability to generate the Convey HC-Series Interface to the Generate Menu under the ROCCC Menu. When selected, the files necessary for the Convey interface will be generated for the file currently selected in the file browser. Generating the Convey interface requires the code first be compiled successfully with ROCCC.

When successful, ROCCC will add a subdirectory named ConveyInterface to the project files associated with this file. Inside the ConveyInterface directory will be the generated hardware wrapper and software that will be used to activate the hardware.

The files generated are the following:

- cae_csr.v



Would you like the generated files to be targeted for simulation?
Yes = Simulation, No = Synthesis

Figure 2: Synthesis versus Simulation

- InferredBRAMFifoXX.v
- Makefile
- microFifo.vhdl
- synchronizationFifo.v
- X_addressGenerators.vhdl
- X.s
- X.v
- XStub.c

When generating the Convey interface, you must specify if the code generated is going to be used for Synthesis or Simulation, as shown in Figure 2. Choose Yes from the pop-up question if you wish to generate code for simulation and No if you wish to generate code for synthesis. The names and types of files generated will be the same.

1.3 Simulating and Synthesizing Convey Code

In order to simulate synthesize ROCCC code for the Convey HC-Series, you will need to set up a directory structure that is expected by the Convey PDK. To do this, you will need a directory with the following subdirectories:

- coregen
- phys
- SampleApp
- sim
- verilog

In order to use the Convey PDK, you will have to set the environment variables CNY_PDK, CNY_PDK_REV, and CNY_PDK_PLATFORM. CNY_PDK should be set to the directory where the latest version of the Convey PDK is installed, typically /opt/convey/pdk. CNY_PDK_REV should be set to the latest revision subdirectory you have installed, for example 2011_11_22. The

CNY_PDK_PLATFORM should be set to hc-1 or hc-1ex depending on the machine you are synthesizing for. Additionally, if you are going to simulate you must set the environment variable CNY_PDK_HDLSIM. If you are using Modelsim, set CNY_PDK_HDLSIM to Mentor.

Place all VHDL and verilog files into the verilog subdirectory. In addition to the files in the ConveyInterface directory, you will require the files in the vhdl directory and any associated VHDL from other ROCCC modules and systems needed.

In order to synthesize, place the Makefile in the phys directory and run make from the phys directory.

If you wish to simulate, you will require additional setup. Copy the contents of the CNY_PDK/CNY_PDK_REV/hc-1/examples/cae_vadd/SampleAppVadd/ directory into your local SampleApp directory. Similarly, copy the contents of the CNY_PDK/CNY_PDK_REV/hc-1/examples/cae_vadd/sim/ directory into your local sim directory. Place the .s (assembly) and .c (C code) files into the SampleApp directory and place the generated Makefile in the root directory renamed Makefile.include. Modify the Makefile in the SampleApp directory to compile the generated .s and .c code instead of the sample UserApp code.

The C code generated by ROCCC is a stub that interfaces with the hardware and can call either the co-simulation environment or the actual hardware. The code, however, is a function that must be called, so it is the user's responsibility to write a main function in C to initialize all data and call the stub function appropriately.

Further, the code that is generated has no way currently of knowing what the nickname should be for your platform. If you are simulating, place the signature ”pdk” in both the .s and .c code at the appropriate place. For synthesis, replace the signature with the nickname associated with your installation.