OpenCPI FPGA Reference Platform, Getting-Started Guide

Shepard Siegel, Atomic Rules LLC (Shepard.Siegel@atomicrules.com)

Revision	Date	By	Notes
0.01	2009-03-10	ssiegel	Draft
0.02	2009-11-08	ssiegel	Update, add Content

This document describes procedures that may be useful in getting started with the OpenCPI FPGA Reference Platform (OC-FRP).

Overview of the Getting-Started Process

The steps below summarize the suggested getting-started sequence:

1. Assemble the hardware components, update, verify
2. Install the Operating System, update, verify
3. Install the FPGA Tools, update, verify
4. Install the OpenCPI Example Codes, build examples, verify

This sequence progressively removes risk and uncertainty. The final step of running OpenCPI example codes is expected to grow over time. The intent is to validate OS, Tools, and FPGA Platform performance in sequence so that once user development commences; the focus will be on IP worker development.

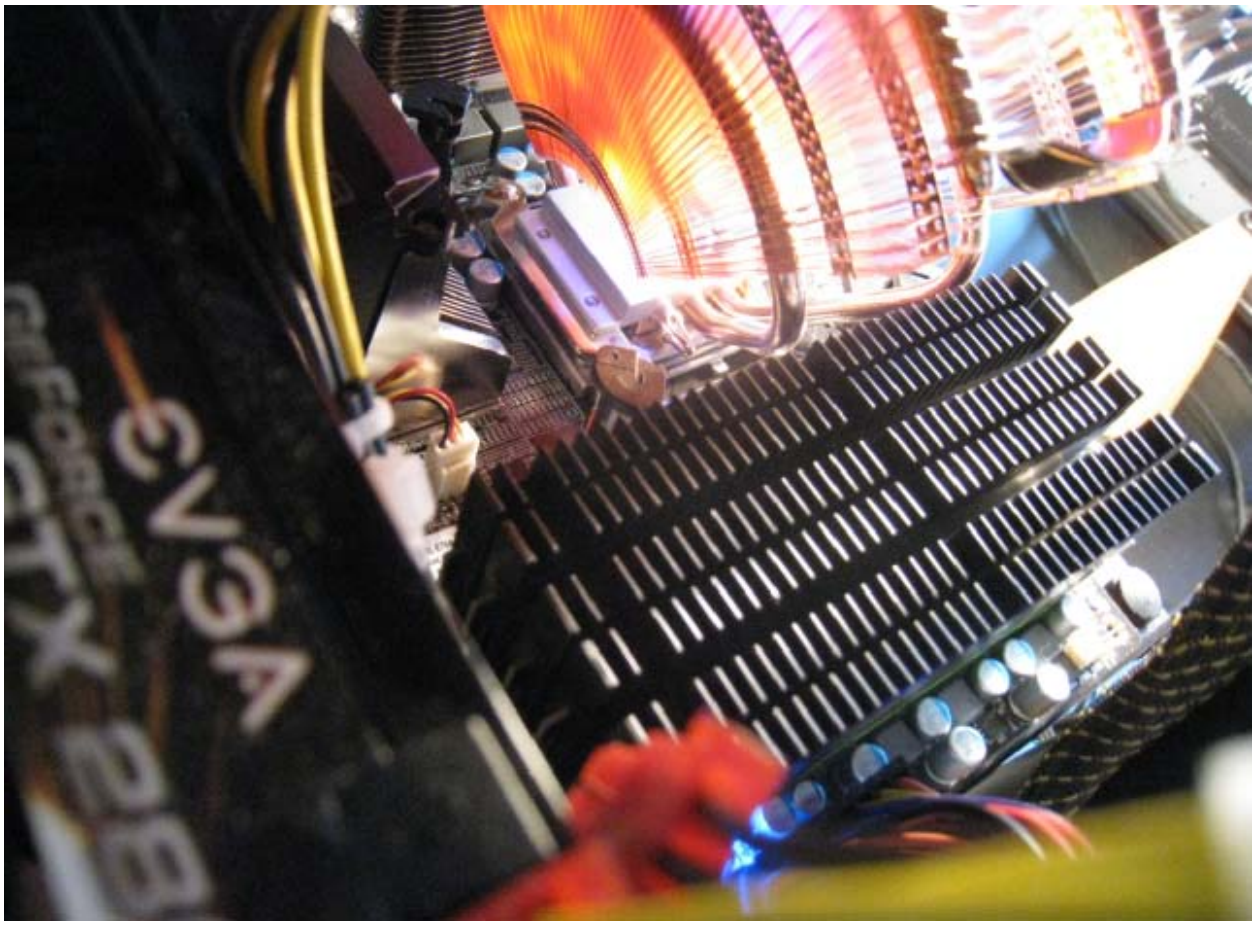
This sequence is not necessarily unique to OpenCPI; and in fact, effort has been made not to require specific versions of hardware and software. Where they are listed, versions and model numbers provide guidance on what has been used and tested. Please help us soften the learning curve for others by providing your questions and feedback.

Hardware Component Assembly

Assemble the computer hardware components in accordance with industry practice for safety and practice. The GPU should be located in the x16 PCIe slot closest to the x58, and the FPGA board in the next x16 PCIe slot.

At this writing (Q4-2009), the corei7 architecture and the x58 Northbridge look ideal for our applications. Specifically the 24 lanes of Gen 2.0 PCIe, split between three PCIe slots, allows a GPU plus two FPGA add in boards. We have mixed and matched the ML555 with the XUPV5 and ML605.

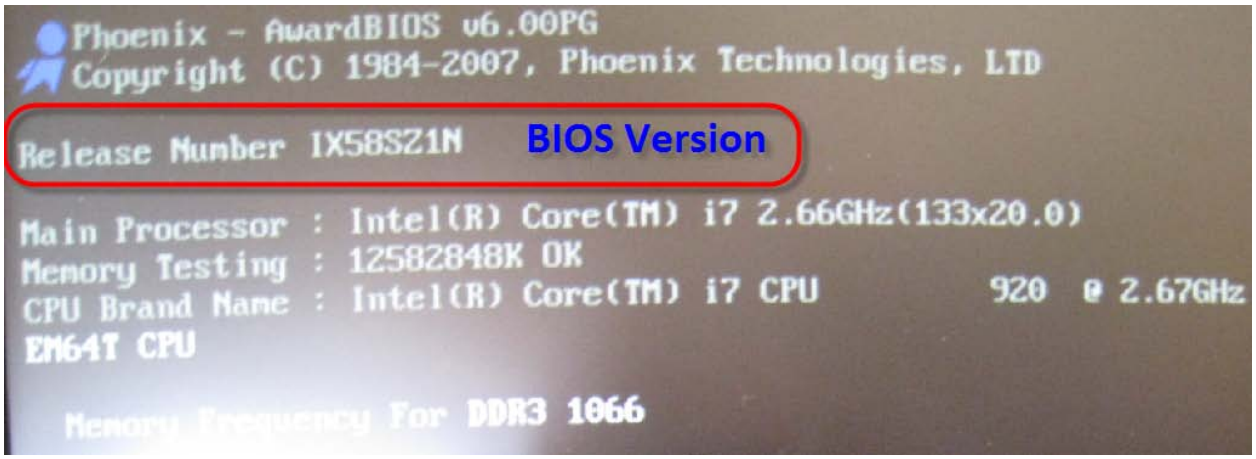
The photo below shows a view of the EVGA x58 motherboard with 6x2GB DDR3 DIMMS installed, as specified in the OpenCPI COTS component system spec. We used a fancy looking fan in this build.



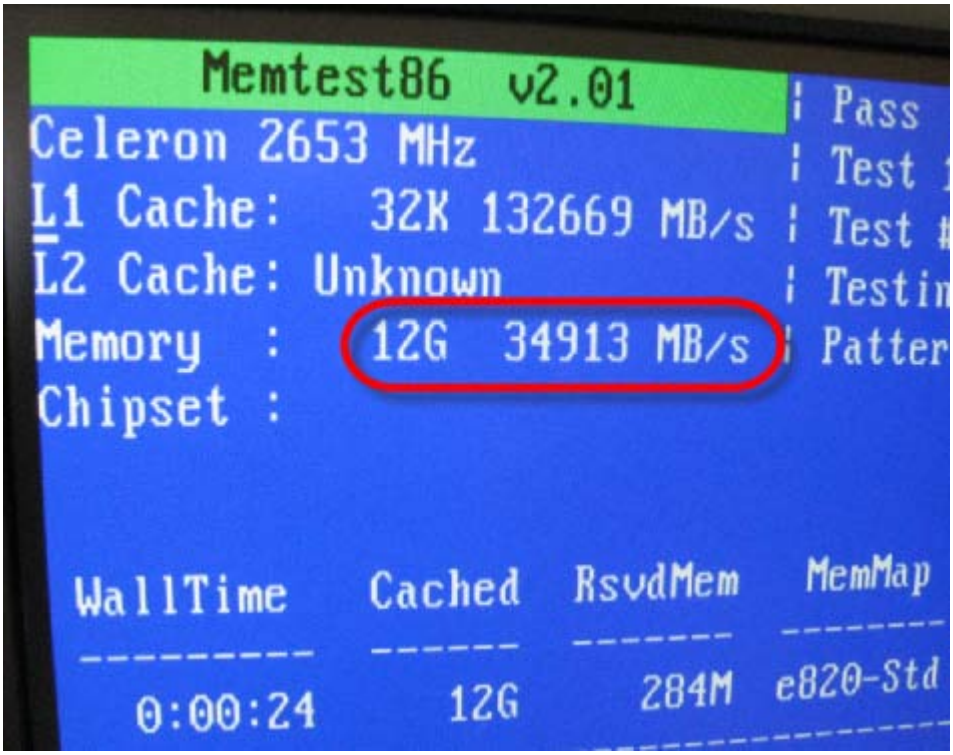
At your discretion, you can save the installation of the FPGA board(s) until after the system is built, the OS installed, and the FPGA Vendor tools installed.

Hardware Component Verification

Upon power-up to the BIOS, observe that the BIOS version is at least version IX58SZ1N, the processor is correctly indentified, and the 12 GB of DDR1066 is detected.



If desired, a memory test, such as from www.memtest.org may be run to further validate memory capacity and bandwidth, prior to installing an OS.



Operating System Installation

Install and update the operating system. Use the updater System→Administration→UpdateManager then "Check" to look for updates.

The specified standard OS is RHEL-WS 64b 5.3. A key factor in selecting this 64b Linux OS is that it is a supported OS by just about every vendor. That said, at Atomic Rules, we have used 64b Ubuntu 8.10 with very few issues for most of our development. The few issues we have had have been related to libraries for vendor FPGA tools, not the core OpenCPI components.

Congratulations, you've built a high-performance Linux box! You may want to think about your backup strategy and related IT issues. Even doing kernel-mode driver work, we yet to lose data when something has gone very wrong and we've hung the OS. (This should never happen to you, the user.) Still, we are a little paranoid and mirror our local disk on a NAS box. We also use a UPS. Use your good judgment.

FPGA Tools Installation

Install and update the FPGA tools. At this writing, we have only used the Xilinx back end tools, as we have only been working with Virtex-5 and Virtex-6 devices to date. The required baseline for the vendor tools for Xilinx is their "ISE 11.3 Logic Edition". This provides not just the usual backend map, place, route; but a first-class simulator (ISim) and first-class RTL synthesis (XST). We have been careful not to *require* the use of 3rd party simulators (like ModelSim) or synthesis (like SynplifyPro); and intend to provide better support for them in the future. The Xilinx tools support both VHDL and Verilog (we simply say "RTL" and assume either).

All of the OpenCPI interfaces that connect workers together are clearly specified as OCP-IP Worker Interface Profiles. These well-defined interfaces are language-agnostic, and work equally well any RTL and the FPGA back end tools.

For development targeting Xilinx FPGAs the only FPGA software required is ISE 11.3-LOGIC. Obtain the software, install it, and if you have a pre-existing RTL design of your own, do some sort of a "smoke test" to up your comfort. Congratulations, now your Linux box has ISE 11.3-LOGIC on it! We will likely advance to future versions of Xilinx ISE as soon as they are publically available.

If you have other FPGA or ESL CAD tools to install; now is good time to that. If your tools produce RTL outputs; now is a good time to test their interaction with the backend tools. For example, Bluespec SystemVerilog ("BSV", or simply "Bluespec") is an ESL that we use in our work. To date, most infrastructure code for OpenCPI is written in BSV. You do not require BSV to use OpenCPI (although we supply both the BSV source code and the compiler's RTL outputs). Because of our standardization on an "RTL Platform", we can mix together most anything that produces an RTL or structural netlist output. For example, Bluespec, C-to-Gates, and RTL; from functional simulation in ISim, to bitstream. Yeah, ISE11.3!

Before the Hardware

A lot can be done without any FPGA hardware. For example, designs can be written, simulated, verified, and debugged; bitstreams built and checked for resources used and timing closure. If you like, read over, but don't perform the "Installing FPGA Boards". Then you can move on to looking at the OpenCPI distribution, run a simulation, build a bitstream, and come back later to actually download and use it.

Installing the FPGA Boards

With a GPU in the PCIe slot closest to the x58; there remain two more PCIe Gen 2 slots. The one closer to the x58 has 16 lanes; the farthest one, only 4. We suggest populating the middle slot first, if you have more than one FPGA board.

When installing the FPGA boards, make sure the switch on the power supply is off so that you don't accidently take the motherboard out of standby. Some boards like the ML555 take all their power from the edge connector. Other boards, like the XUPV5 and ML605 require a 4-pin Molex connector. We recommend using the workstation power supply (e.g. the Enermax EMD625AWD) to feed the Molex connector that supplies power to the FPGA board. DO NOT USE both the 4-pin Molex AND the wall wart. Reread the user guide that came with the FPGA board if any questions.

Check all jumper, switch, and dip-switch settings if you have not already done so. You will need to make some board-specific choices as to which non-volatile storage will supply the bitstream at power-up.

Plug in your USB/JTAG bridge. You will use one of these for each FPGA board. This is how you will access the JTAG chain on the board. JTAG is used for many purposes including, directly or indirectly reaching non-volatile "FLASH" storage, writing bitstreams to the FPGA directly, talking to ChipScope and other JTAG-accessible cores.

At this point, you should be able to boot up the workstation with the FPGA board(s) in them. Most boards have a benign bitstream in them as shipped from the factory. This means there probably is not a PCIe core in the bitstream that will boot at power up. This is fine, we will be able to boot the PC, but the BIOS won't "see" the board. We will fix that later.

What you should be able to do is use the Xilinx tool "Impact" to open your USB Cable and observe the JTAG chain. This is a serial chain of devices including FLASH and, of course, the FPGA.

At this point you pretty much have everything you need to build and recreate some OpenCPI applications.

The OpenCPI RPL Distribution

Instantaneous snapshots of OpenCPI RPL distribution are commonly sent as gzipped tarballs in the form of "ocpi-YYYYMMDD_HHMM.tar.-gz". The uppercase letters containing the date and time the tar was created. The ".-gz" suffix to allow the gzip past spam filters, such as Gmail's.

The directory structure is not finalized; this is a work in progress.

The doc dir contains this and other documentation. In specific, the OCP-IP Worker Interface Profiles (WIP) functional spec is there. The "WIP Spec" describes the mapping from WIP Attributes to OCP-IP parameters. And the OCP-IP Spec (available at www.ocpip.org) provides the mapping from parameters to signals. In this way, OpenCPI interfaces are a proper-subset of a standard core-core interconnect.

We have endeavored to use command-line, scriptable approaches to all builds. There is a Makefile at the root with multiple targets. To kick off the transformation of RTL to bitstream, you can say

```
% make application-target
```

Where application is the name of an OpenCPI application and target is the board. We are still working this out, but because of the FTop/CTop structure, very little (if anything) in the application changes when moving to another device. In fact we have ported the DMA loopback application "oc1001" to the ML555, XUPV5, and ML605 with no changes in the application CTop at all.

Have 30 minutes? Open a terminal window and change directory to the root of the ocpi disti and type

```
% make oc1001-ml555
```

(or `–xupv5`, or `–ml605`, etc). This will run the sequence of operations to transform the RTL to bitstream.

When the bitstream has been built, it can be loaded to the FPGA on board with "loadBitstream".

Build and Test OpenCPI Example Codes

For most examples the following steps are required:

1. Change directory to $OPENCPI/examples/<example_name>
2. Run the functional simulation script – a short, directed test
3. Run the bitstream build script – (Synthesis, Map, Place, Route, Bitgen)
4. Transfer the bitstream from the computer to the FPGA using "loadBitstream"
5. Run the example (typically a host command in the example directory)

We have a technique that reduces step 4 to about ten seconds and does not require a reboot of the computer. It does require, however, that prior to BIOS scan at powerup, the FPGA has been loaded with an OpenCPI boot bitstream. The process to place this OpenCPI boot bitstream in non-volatile storage is described separately.