OpenCPI
ML605 Hardware Setup Guide

Version 1.4

Revision History

Revision	Description of Change	Date
v1.1	Initial Release	3/2017
v1.2	Updated for OpenCPI Release 1.2	8/2017
v1.3	Updated for OpenCPI Release 1.3	1/2018
v1.3.1	Updated for OpenCPI Release 1.3.1	3/2018
v1.4	Updated document format and directory paths	9/2018

Table of Contents

1 ML605 Hardware Setup **4**
1.1 Prerequisites . . . 4
1.2 Hardware Description . . . 4
1.3 Setup Overview . . . 4
1.3.1 Configure the board to boot from flash memory and the PCIe lane size . . . 5
1.3.2 Install board into OpenCPI development host . . . 6
1.3.3 Verify JTAG connection using OpenCPI script . . . 6
1.3.4 Run OpenCPI loadFlash script . . . 6
1.3.5 Verify that the ML605 is configured with a OpenCPI bitstream . . . 7
1.3.6 Modification of /opt/opencpi/system.xml . . . 8

1 ML605 Hardware Setup

1.1 Prerequisites

- A development system which has Xilinx ISE version 14.7 installed. While OpenCPI has been shown to work with older versions of ISE, these instructions are performed using version 14.7.
- Xilinx cable drivers installed. Required for OpenCPI's loadFlash command.
- An installed OpenCPI framework from source or RPMs.
- The OpenCPI provided projects (core and assets) have been built, by the user, for the *ml605* platform.

1.2 Hardware Description

The ML605 is the PCI-Express development board for Virtex 6. It can be purchased directly from Xilinx or a distributor. Xilinx's URL for the board containing user guides and reference manuals can be found here:

```
http://www.xilinx.com/products/boards-and-kits/ek-v6-ml605-g.html#documentation
```

There is one version of the board. The below table contains platform name and part number for the development board.

OpenCPI Platform Name	Xilinx Kit Part Number	FPGA Part Number
ml605	EK-V6-ML605-G	XC6VLX240T-1FFG1156

1.3 Setup Overview

To use this board within OpenCPI, the board must be configured to load the FPGA with an OpenCPI bitstream upon power up. There is flash memory on the board which can be loaded with a bitstream to load the FPGA on power up. The process for loading this flash is:

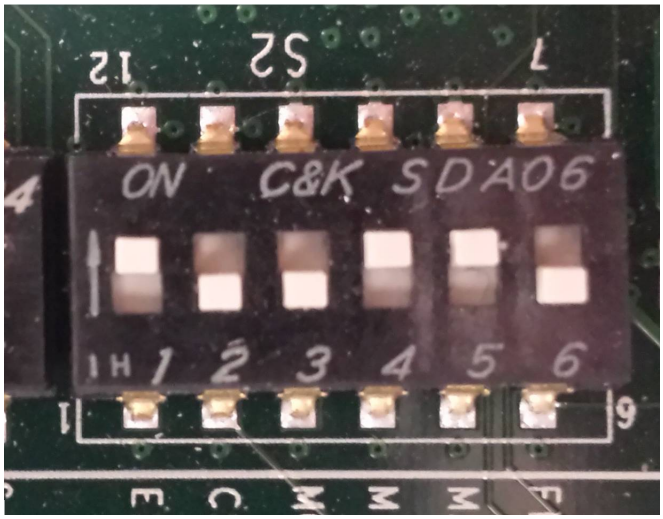
1. Configure the board to boot from flash memory and the PCIe lane size.
2. Install board into OpenCPI development host.
3. Verify JTAG connection using OpenCPI script.
4. Run OpenCPI loadFlash script.
5. Verify that the ML605 is configured with a OpenCPI bitstream.
6. Modification of /opt/opencpi/system.xml

Details for implementing this procedure can be found in the sections below.

1.3.1 Configure the board to boot from flash memory and the PCIe lane size

For more information about configuring the ML605, reference Xilinx ML605 Hardware User Guide, UG534.pdf (v1.8) October 2, 2012.

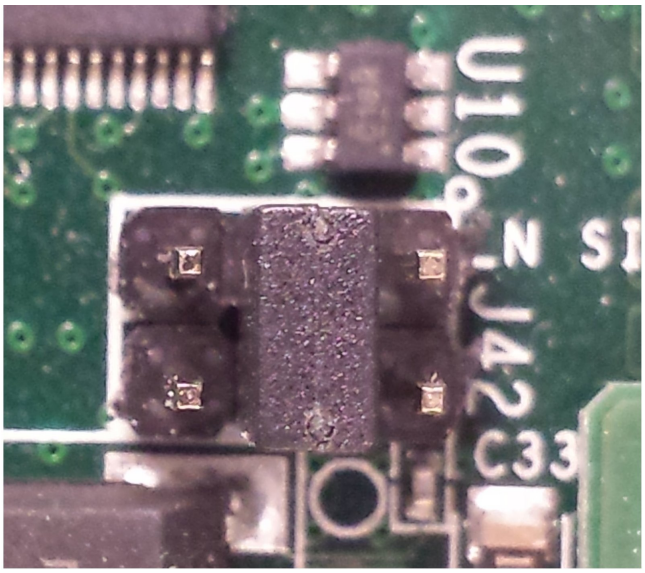
1. The S2 DIP switch is a multi-purpose selector switch. It controls the state of the FPGA's configuration mode pins and must be set for "Slave SelectMAP Platform Flash XL" to boot from flash memory. Below is a picture of the S2 DIP switch with the expected setting:



The table below contains the expected S2 DIP switch settings.

S2.1	S2.2	S2.3	S2.4	S2.5	S2.6
on	off	off	on	on	don't care

2. The J42 configures the board to support 4x PCIe lanes. Below is a picture of J42 with a 100mil shunt installed in the correct position.



1.3.2 Install board into OpenCPI development host

1. Power off the development host.
2. Following anti-static best practices, insert the ML605 into a compatible PCIe slot.
3. Attach the USB cable between the ML605 JTAG port and a host USB port.
4. With the AUX power supply unplugged (powered OFF), attach the AUX power cable to the ML605 AUX power-in connector.
5. Power ON the AUX power supply by plugging it an A/C outlet. (at this point the development host is still OFF)
6. Locate the SW2 on the ML605, and switch it to the ON position, i.e, slide away from the AUX power-in connector. The board will power up, but the development host is not powered on.
7. Power ON the development host.

1.3.3 Verify JTAG connection using OpenCPI script

1. Open a terminal configured for OpenCPI and run the following script to test the JTAG connection:

   ```
   $ probeJtag
   ```

   Example of the output is shown below:

   ```
   ==== Probing for Altera JTAG ports:
   Altera directory set by (OCPI_ALTERA_TOOLS_DIR) does not exist.
   ==== Probing for Xilinx JTAG ports:
   Discovered ports are: usb21
   Trying port usb21...  ESN is 000013C8121101
   Part at position 1 on is xccace
   Part at position 2 on is xc6vlx240t.
   ```

   For the purposes of this guide, messages related to Altera may be ignored. If the output doesn't look like this, refer Xilinx documentation to ensure the JTAG cable is correctly installed and operational.

1.3.4 Run OpenCPI loadFlash script

1. The loadFlash script is located at /opt/opencpi/cdk/scripts. You must pass it a pre-built ML605 .bitz file generated by OpenCPI and the serial number (in hex) of the JTAG cable from probeJtag script above. Programming the flash memory with a bitstream takes approximately 10 minutes to complete.

   ML605 syntax

   ```
   $  loadFlash ml605 <OpenCPI Bitstream File> <JTAG_CABLE_SERIAL_NUMBER>
   ```

   Example command and output:

   ```
   $ loadFlash ml605 hdl/platforms/ml605/testbias_ml605_base.bitz 000013C8121101

   Loading the flash memory on the ml605 platform attached to the JTAG pod with ESN 000013C8121101
   Loading from file: hdl/platforms/ml605/testbias_ml605_base.bitz
   Looking for the Xilinx USB port corresponding to JTAG Pod ESN 000013C8121101.
   Discovered ports are: usb21
   Trying usb21... ESN is: 000013C8121101
   Found ESN 000013C8121101 on usb21, using it.
   The bitstream file "hdl/platforms/ml605/testbias_ml605_base.bitz" is compressed. Expanding it.
   Bitstream file decompressed into "/tmp/ocpibitstream3100.bit"
   ```

```
Converting "/tmp/ocpibitstream3100.bit" to flash format in
"/tmp/ocpibitstream3100.mcs" using xilinx promgen tool.
Executing command: /opt/Xilinx/14.5/ISE_DS/ISE/bin/lin64/promgen -w -p mcs -c FF
-x xcf128x -data_width 16 -u 00000000 /tmp/ocpibitstream3100.bit
Conversion to flash file format succeeded. Starting flash programming.
This can take a while - 10-15 minutes. Starting at 09:57:27.
real 11m55.080s
user 0m28.494s
sys 0m7.742s
Flash programming is complete. You must power-cycle the system to use it
Use the "ocpihdl search" command after power cycling to confirm success.
```

2. Once programming is complete, power cycle the system:

i Power OFF the host

ii Power cycle the ML605

iii Power ON the host

1.3.5 Verify that the ML605 is configured with a OpenCPI bitstream

1. Open a terminal window.

2. Execute the following Linux command to verify the presence of the ML605.

```
$ sudo lspci -vv

03:00.0 RAM memory: Xilinx Corporation Device 4243 (rev 02)
Subsystem: Xilinx Corporation Device 0007
Control: I/O- Mem- BusMaster- SpecCycle- MemWINV- VGASnoop- ParErr- Stepping-
SERR- FastB2B- DisINTx-
Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort-
<TAbort- <MAbort- >SERR- <PERR- INTx-
Interrupt: pin A routed to IRQ 11
Region 0: Memory at e8000000 (32-bit, non-prefetchable) [disabled] [size=64M]
Region 1: Memory at ec000000 (32-bit, non-prefetchable) [disabled] [size=1M]
Capabilities: [40] Power Management version 3
Flags: PMEClk- DSI- D1- D2- AuxCurrent=0mA PME(D0+,D1+,D2+,D3hot+,D3cold-)
Status: D0 NoSoftRst+ PME-Enable- DSel=0 DScale=0 PME-
Capabilities: [48] MSI: Enable- Count=1/1 Maskable- 64bit+
Address: 0000000000000000  Data: 0000
Capabilities: [60] Express (v2) Endpoint, MSI 01
DevCap: MaxPayload 512 bytes, PhantFunc 0, Latency L0s <64ns, L1 unlimited
ExtTag- AttnBtn- AttnInd- PwrInd- RBE+ FLReset-
DevCtl: Report errors: Correctable- Non-Fatal- Fatal- Unsupported-
RlxdOrd- ExtTag- PhantFunc- AuxPwr- NoSnoop+
MaxPayload 256 bytes, MaxReadReq 256 bytes
DevSta: CorrErr- UncorrErr- FatalErr- UnsuppReq- AuxPwr- TransPend-
LnkCap: Port \#0, Speed 5GT/s, Width x4, ASPM L0s, Latency L0 unlimited, L1 unlimited
ClockPM- Surprise- LLActRep- BwNot-
LnkCtl: ASPM Disabled; RCB 64 bytes Disabled- Retrain- CommClk-
ExtSynch- ClockPM- AutWidDis- BWInt- AutBWInt-
LnkSta: Speed 5GT/s, Width x4, TrErr- Train- SlotClk- DLActive- BWMgmt- ABWMgmt-
DevCap2: Completion Timeout: Range B, TimeoutDis-, LTR-, OBFF Not Supported
DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis-, LTR-, OBFF Disabled
LnkCtl2: Target Link Speed: 5GT/s, EnterCompliance- SpeedDis-
 Transmit Margin: Normal Operating Range, EnterModifiedCompliance- ComplianceSOS-
 Compliance De-emphasis: -6dB
```

```
LnkSta2: Current De-emphasis Level: -3.5dB, EqualizationComplete-, EqualizationPhase1-
 EqualizationPhase2-, EqualizationPhase3-, LinkEqualizationRequest-
Capabilities: [100 v1] Device Serial Number 00-00-00-01-01-00-0a-35
Kernel modules: windrvr6
```

3. In the terminal window configured for OpenCPI, execute the OpenCPI utility that executes Xilinx iMPACT to scan JTAG for devices:

```
$ probeJtag

==== Probing for Altera JTAG ports:
Altera directory (set by OCPI_ALTERA_TOOLS_DIR) does not exist.
==== Probing for Xilinx JTAG ports:
Discovered ports are: usb21
Trying port usb21...  ESN is 000013C8121101
Part at position 1 on is xccace
Part at position 2 on is xc6vlx240t
```

4. Load the OpenCPI driver (requires sudo privileges):

```
$ ocpidriver load
```

5. Execute an OpenCPI utility

```
$ ocpihdl search
OpenCPI HDL device found: 'PCI:0000:03:00.0': bitstream date Fri Jun 26 15:14:21 2015,
platform "ml605", part "xc6vlx240t", UUID 8cd0a91e-1c37-11e5-8b5f-f30e2f82978a
```

6. Perform a container check using ocpirun -C. The result should look like this:

```
$ ocpirun -C
Available containers:
 #  Model Platform    OS     OS Version  Arch   Name
 0  rcc   centos7     linux  c7          x86_64 rcc0
 1  hdl   ml605                                PCI:0000:03:00.0
```

1.3.6 Modification of /opt/opencpi/system.xml

When multiple boards from same vendor are installed, and thus more than one JTAG USB, the OpenCPI load function (either explicit in "ocpihdl load" or implicit in 'ocpirun' needs to know which JTAG USB to use. The '/opt/opencpi/system.xml' file is used to map the PCI:* with JTAG USB serial number information. An example is shown below:

```
<opencpi>
    <container>
        <hdl>
        <device name="0000:03:00.0" esn="000013C8121101"/>
        <device name="0000:08:00.0" esn="000013C84C2F01"/>
        </hdl>
    </container>
</opencpi>
```

Perform a container check using *ocpirun -C*. The result should look like this:

```
$ ocpirun -C
Available containers:
 #  Model Platform    OS       OS Version  Arch   Name
 0  rcc   centos7     linux    c7          x86_64 rcc0
 1  hdl   ml605                                  PCI:0000:08:00.0
 2  hdl   ml605                                  PCI:0000:03:00.0
```