

ALST4 Getting Started Guide

1 Hardware Prerequisites

This section describes the hardware prerequisites required for an operational alst4 (Altera Stratix IV) platform using OpenCPI. The optional HSMC Debug Loopback and HSMC Debug Breakout Header cards are only intended for testing purposes. Also note that the slot configurations in Table 1 are limited by what FPGA bitstreams are currently built by OpenCPI and not by what hardware configurations are theoretically possible using OpenCPI.

Hardware prerequisites are as follows.

- A Stratix IV GX230 board, which has undergone an OpenCPI-specific initial one-time hardware setup [1] and is plugged into a PCIE slot of an x86 computer.
- Optionally, one of the following HSMC card configurations in Table 1 may exist

Table 1: OpenCPI-supported Stratix IV hardware HSMC slot configurations

	HSMC A slot	HSMC B slot
Test loopback A setup	HSMC Debug Loopback Card	(empty)
Test loopback B setup	(empty)	HSMC Debug Loopback Card
Test dual loopback setup	HSMC Debug Loopback Card	HSMC Debug Loopback Card
Test breakout A setup	HSMC Debug Breakout Header Card	(empty)
Test breakout B setup	(empty)	HSMC Debug Breakout Header Card
Zipper A setup	Modified[4] Zipper/MyriadRF transceiver card	(empty)
Zipper B setup	(empty)	Modified[4] Zipper/MyriadRF transceiver card

2 Software Prerequisites

- A CentOS 6 or CentOS 7 operating system installed on the x86 computer.
- Altera Quartus installed on the x86 computer. For more information refer to [2]
- OpenCPI framework and prerequisite RPMs installed on the x86 computer. For more information refer to [3]
- OpenCPI core project compiled for alst4.
- OpenCPI assets project compiled for alst4.

3 Driver Setup

If you want to use more than 128KB of RAM, then you will need to reserve a block of memory during the Linux kernel boot, using the memmap parameter. The memmap parameter takes a number of formats, but the one that is most useful to us is the following:

```
memmap=SIZE$START
```

Where SIZE is the number of bytes to reserve in either hex or decimal, and START is the physical address in hexadecimal bytes. You *must* use even page boundaries (0x1000 or 4096 bytes) for all addresses and sizes.

3.1 Calculate Values in Preparation for Memory Reservation

Start by running:

```
dmesg | grep BIOS
```

The output will look something like:

```
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009f800 (usable)
 BIOS-e820: 000000000009f800 - 00000000000a0000 (reserved)
 BIOS-e820: 00000000000ca000 - 00000000000cc000 (reserved)
 BIOS-e820: 00000000000dc000 - 00000000000e4000 (reserved)
 BIOS-e820: 00000000000e8000 - 00000000000100000 (reserved)
 BIOS-e820: 00000000000100000 - 00000000005fef0000 (usable)
 BIOS-e820: 00000000005fef0000 - 00000000005feff000 (ACPI data)
 BIOS-e820: 00000000005feff000 - 00000000005ff00000 (ACPI NVS)
 BIOS-e820: 00000000005ff00000 - 000000000060000000 (usable)
 BIOS-e820: 000000000060000000 - 0000000000f0000000 (reserved)
 BIOS-e820: 0000000000fec00000 - 0000000000fec10000 (reserved)
 BIOS-e820: 0000000000fee00000 - 0000000000fee01000 (reserved)
 BIOS-e820: 0000000000ffe0000 - 0000000000100000000 (reserved)
```

You want to select a (usable) section of memory and reserve a section of that memory. Once the memory is reserved, the Linux kernel will ignore it. In this example, there are 3 useable sections:

```
BIOS-e820: 0000000000000000 - 000000000009f800 (usable)
BIOS-e820: 00000000000100000 - 00000000005fef0000 (usable)
BIOS-e820: 00000000005ff00000 - 000000000060000000 (usable)
```

Due to the way Linux manages memory, it is recommended you pick a higher address (above the first 24 bits). The best choice is the second section (pages 0x100-0x5fef0). If you wanted to reserve 128MB, that would be 0x8000 pages. Pick the end of the block (page 0x5fef0) and subtract the number of pages, leaving 0x57ef0. This would result in the following memmap parameter:

```
memmap=128M$0x57EF0000
```

3.2 Configure Memory Reservation

Once you've calculated your memmap parameter, you will need to add it to the kernel command line in your boot loader.

For CentOS, you can use the utility "grubby".

This will add the parameter to all kernels in the startup menu. The single quotes are REQUIRED or your shell will interpret the \$0:

For *CentOS6*:

```
sudo grubby --update-kernel=ALL --args=memmap='128M$0x57EF0000'
```

CentOS 7 uses grub2, which requires a double backslash to not interpret it:

```
sudo grubby --update-kernel=ALL --args=memmap='128M\\$0x57EF0000'
```

To verify the current kernel has the argument set:

```
sudo -v
sudo grubby --info $(sudo grubby --default-kernel)
```

CentOS 7 users should see a SINGLE backslash before the \$, for example:

```
args="ro rdblacklist=nouveau crashkernel=auto rd.lvm.lv=vg.0/root quiet audit=1 boot=UUID=96933\
cb5-f478-4933-a0d4-16953cf47f5c memmap=128M\0x57EF0000 LANG=en_US.UTF-8"
```

If no longer desired, the parameter can also be removed:

```
sudo grubby --update-kernel=ALL --remove-args=memmap
```

More information concerning grubby can be found at:

https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/sec-Making_Persistent_Changes_to_a_GRUB_2_Menu_Using_the_grubby_Tool.html ...

the memmap parameter:

<https://www.kernel.org/doc/html/latest/admin-guide/kernel-parameters.html>

Note: If you have other memmap parameters, e.g. for non-OpenCPI PCI cards, then grubby usage will be different. The OpenCPI driver will use the first memmap parameter on the command line OR the parameter “opencpi_memmap” if it is explicitly given. If this parameter is given, the standard memmap command with the same parameters must ALSO be passed to the kernel.

3.3 Apply Memory Reservation

Reboot the system, making certain to boot from your new configuration.

3.4 Verify Memory Reservation

Once that’s done, if you run ‘dmesg’ you should see something like this:

```
dmesg | more
Linux version 2.6.18-128.el5 (mockbuild@hs20-bc1-7.build.redhat.com) (gcc version 4.1.2 \
20080704 (Red Hat 4.1.2-44)) #1 SMP Wed Dec 17 11:41:38 EST 2008
Command line: ro root=/dev/VolGroup00/LogVol00 rhgb quiet memmap=128M0x57EF0000
BIOS-provided physical RAM map:
 BIOS-e820: 0000000000000000 - 000000000009f800 (usable)
 BIOS-e820: 000000000009f800 - 00000000000a0000 (reserved)
 BIOS-e820: 00000000000ca000 - 00000000000cc000 (reserved)
 BIOS-e820: 00000000000dc000 - 00000000000e4000 (reserved)
 BIOS-e820: 00000000000e8000 - 00000000000100000 (reserved)
 BIOS-e820: 00000000000100000 - 00000000005fef0000 (usable)
 BIOS-e820: 00000000005fef0000 - 00000000005feff000 (ACPI data)
 BIOS-e820: 00000000005feff000 - 0000000000ff00000 (ACPI NVS)
 BIOS-e820: 0000000000ff00000 - 000000000060000000 (usable)
 BIOS-e820: 000000000060000000 - 0000000000f0000000 (reserved)
 BIOS-e820: 0000000000fec00000 - 0000000000fec10000 (reserved)
 BIOS-e820: 0000000000fee00000 - 0000000000fee01000 (reserved)
 BIOS-e820: 0000000000fffe0000 - 0000000000100000000 (reserved)
user-defined physical RAM map:
 user: 0000000000000000 - 000000000009f800 (usable)
 user: 000000000009f800 - 00000000000a0000 (reserved)
 user: 00000000000ca000 - 00000000000cc000 (reserved)
 user: 00000000000dc000 - 00000000000e4000 (reserved)
 user: 00000000000e8000 - 00000000000100000 (reserved)
 user: 00000000000100000 - 000000000057ef0000 (usable)
 user: 000000000057ef0000 - 00000000005fef0000 (reserved) <== New
 user: 00000000005fef0000 - 00000000005feff000 (ACPI data)
```

```

user: 000000005feff000 - 000000005ff00000 (ACPI NVS)
user: 000000005ff00000 - 0000000060000000 (usable)
user: 00000000e0000000 - 00000000f0000000 (reserved)
user: 00000000fec00000 - 00000000fec10000 (reserved)
user: 00000000fee00000 - 00000000fee01000 (reserved)
user: 00000000fffe0000 - 0000000100000000 (reserved)
DMI present.

```

You will see a new (reserved) area between the second (useable) section and the (ACPI data) section.

Now, when you run the 'make load' script, it will detect the new reserved area, and pass that data to the opencpi kernel module.

4 Driver Notes

When available, the driver will attempt to make use of the CMA region for direct memory access. In use cases where many memory allocations are made, the user may receive the following kernel message:

```
alloc_contig_range test_pages_isolated([memory start], [memory end]) failed
```

This is a kernel warning, but does not indicate that a memory allocation failure occurred, only that the CMA engine could not allocate memory in the first pass. Its default behavior is to make a second pass and if that succeeded the end user should not see any more error messages. An actual allocation failure will generate unambiguous error messages.

5 Proof of Operation

The following commands may be run in order to verify correct OpenCPI operation on the x86/Stratix IV system.

Existence of alst4 RCC/HDL containers may be verified by running the following command and verifying that similar output is produced.

```

ocpirun -C
Available containers:
#  Model Platform    OS      OS-Version Arch    Name
0  rcc  centos7         linux  c7      x86_64  rcc0
1  hdl  alst4
                                PCI:0000:02:00.0

```

Operation of the RCC container can be verified by running the hello application via the following command and verifying that identical output is produced. Note that the OCPI_LIBRARY_PATH environment variable must be setup to include the hello_world.rcc built shared object file prior to running this command.

```

ocpirun -t 1 assets/applications/hello.xml
Hello, world

```

Simultaneous RCC/HDL container operation can be verified by running the testbias application via the following command and verifying that identical output is produced. Note that the OCPI_LIBRARY_PATH environment variable must be setup correctly for your system prior to running this command.

```

ocpirun -d -m bias=hdl assets/applications/testbias.xml
Property 0: file_read.fileName = "test.input" (cached)
Property 1: file_read.messagesInFile = "false" (cached)
Property 2: file_read.opcode = "0" (cached)
Property 3: file_read.messageSize = "16"

```

```

Property 4: file_read.granularity = "4" (cached)
Property 5: file_read.repeat = "<unreadable>"
Property 6: file_read.bytesRead = "0"
Property 7: file_read.messagesWritten = "0"
Property 8: file_read.suppressEOF = "false"
Property 9: file_read.badMessage = "false"
Property 10: file_read.ocpi_debug = "false" (parameter)
Property 11: file_read.ocpi_endian = "little" (parameter)
Property 12: bias.biasValue = "16909060" (cached)
Property 13: bias.ocpi_debug = "false" (parameter)
Property 14: bias.ocpi_endian = "little" (parameter)
Property 15: bias.test64 = "0"
Property 16: file_write.fileName = "test.output" (cached)
Property 17: file_write.messagesInFile = "false" (cached)
Property 18: file_write.bytesWritten = "0"
Property 19: file_write.messagesWritten = "0"
Property 20: file_write.stopOnEOF = "true" (cached)
Property 21: file_write.ocpi_debug = "false" (parameter)
Property 22: file_write.ocpi_endian = "little" (parameter)
Property 3: file_read.messageSize = "16"
Property 5: file_read.repeat = "<unreadable>"
Property 6: file_read.bytesRead = "4000"
Property 7: file_read.messagesWritten = "251"
Property 8: file_read.suppressEOF = "false"
Property 9: file_read.badMessage = "false"
Property 15: bias.test64 = "0"
Property 18: file_write.bytesWritten = "4000"
Property 19: file_write.messagesWritten = "250"

```

Known Issues

JTAG Daemon

When loading FPGA bitstreams onto the alst4 FPGA (which can occur when running either `ocpihdl load` or `ocpirun`), multiple issues exists with the Altera jtag daemon which may cause the FPGA loading to fail. The following is an example of the terminal output when this failure occurs:

```

Checking existing loaded bitstream on OpenCPI HDL device "PCI:0000:0b:00.0"...
OpenCPI FPGA at PCI 0000:0b:00.0: bitstream date Wed Oct 19 16:04:45 2016, platf
orm "alst4", part "ep4sgx230k", UUID 482195b4-9637-11e6-8002-d76b7b3cbb11
Existing loaded bitstream looks ok, proceeding to snapshot the PCI configuration
(into /tmp/ocpibitstream15980.1).
Scanning for JTAG cables...
Found cable "USB-Blaster [3-11]" to use for device "PCI:0000:0b:00.0" (no serial
number specified).
Error: did not find part ep4sgx230k in the jtag chain for cable USB-Blaster [3-1
1].
Look at /tmp/ocpibitstream15980.log for details.
Error: Could not find jtag position for part ep4sgx230k on JTAG cable "USB-Blast
er [3-11]".
OpenCPI FPGA at PCI 0000:0b:00.0: bitstream date Wed Oct 19 16:04:45 2016, platf
Exception thrown: Bitstream loading error (exit code: 1) loading "../hdl/ass
emblies/dc offset iq imbalance mixer cic dec timestamp/containers-dc offset iq imba
lance mixer cic dec timestamp alst4 base alst4 adc hsmc port b/target-stratix4/dc
offset iq imbalance mixer cic dec timestamp alst4 base alst4 adc hsmc port b.sof.gz
" on HDL device "PCI:0000:0b:00.0" with command: /opt/opencpi/cdk//scripts/loadBi
tStream "../hdl/assemblies/dc offset iq imbalance mixer cic dec timestamp/conta

```

```
iner-dc offset iq imbalance mixer cic dec timestampers alst4 base alst4 adc hsmc port b
/target-stratix4/dc offset iq imbalance mixer cic dec timestampers alst4 base alst4 adc
hsmc port b.sof.gz" "PCI:0000:0b:00.0" "alst4" "ep4sgx230k" "" ""
```

The failure may also manifest as a permissions issue:

```
Scanning for JTAG cables...
JTAG cable setup for platform "alst4" failed.
Dump of /tmp/ocpibitstream5904.cables:
*****
Cable "USB-Blaster variant [3-7]": cannot get serial number.
*****
Dump of /tmp/ocpibitstream5904.log:
*****
Error when locking chain - Insufficient port permissions
*****
```

The follow commands implement a known remedy for each of the aforementioned errors:

```
sudo killall jtagd
sudo chmod 755 /sys/kernel/debug/usb/devices
sudo chmod 755 /sys/kernel/debug/usb
sudo chmod 755 /sys/kernel/debug
sudo mount --bind /dev/bus /proc/bus
sudo ln -s /sys/kernel/debug/usb/devices /proc/bus/usb/devices
sudo <quartus_directory>/bin/jtagd
sudo <quartus_directory>/bin/jtagconfig
```

Single Port of Data from CPU to FPGA

The current implementations of the PCI-e Specification on the this platform only correctly implements data flow from the CPU to the FPGA under certain configurations which must be met when defining new Assemblies:

- At most a single data port with CPU-to-FPGA data flow. Port connection must also be one of:
 1. defined in a single-worker Assembly XML using the worker “`Externals='true'`” attribute/value and the DefaultContainer used (DefaultContainer not defined in Assembly Makefile), or
 2. the first External Assembly Connection defined in the Assembly XML and the DefaultContainer used, or
 3. the first Interconnect Container Connection defined in a Container XML (Default Container must be disabled via “`DefaultContainer=`” in the Assembly Makefile).

Note that this applies to the TX/DAC data path connections for bitstreams with transceiver transmit data flow from a CPU (e.g. RCC worker to FPGA TX/DAC data path). See `projects/assets/hdl/assemblies/empty/cnt_1rx_1tx_bypassasm_fmcomms_2_3_1pc_LVDS_m1605.xml` as an example.

References

- [1] ALST4 Hardware Setup
https://opencpi.github.io/assets/alst4_hardware_setup.pdf
- [2] FPGA Vendor Tools Guide
https://opencpi.github.io/FPGA_Vendor_Tools_Installation_Guide.pdf
- [3] OpenCPI RPM Installation Guide
https://opencpi.github.io/RPM_Installation_Guide.pdf
- [4] Required Modifications for Myriad-RF 1 and Zipper Daughtercards
https://opencpi.github.io/assets/Required_Modifications_for_Myriad-RF_1_Zipper_Daughtercards.pdf