Ashling

Opella-LD Debug Probe

User Manual
v0.8

Trademark Acknowledgements

Ashling is a registered trademark of Ashling Microsystems Limited.

All other trademarks referred to herein are the property of their respective owners.

Ashling
Lonsdale Road, NTP,
Limerick City, Ireland
V94 W9FP

Telephone +353-61-334466
Fax +353-61-334477
Email **support@ashling.com**
Web **http://www.ashling.com**

Disclaimer

Ashling makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability of fitness for any particular purpose. Further, Ashling reserve the right to revise this publication and its hardware peripheral from time to time, without obligation to notify any person of such revision or changes.

Version

v0.8, 29th March 2022
Source File Name: Opella-LD_User_Manual.doc

CONTENTS

Chapter 1. Introduction to the Ashling Opella-LD Debug Probe

1.1 Introduction

Ashling's Opella-LD Debug Probe as shown in Figure 1 is a powerful JTAG/SWD/cJTAG Debug Probe for embedded development with several different target architectures including Synopsys ARC, Arm and RISC-V powered systems.

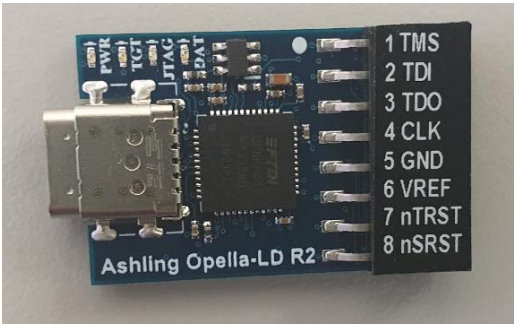

Figure 1. The Ashling Opella-LD Debug Probe

Opella-LD software debug support is provided via the open-source OpenOCD (http://openocd.org/doc/html/About.html) standard and provides fast code download to the target system with control and interrogation support for all core-processor and system resources including registers and memory.

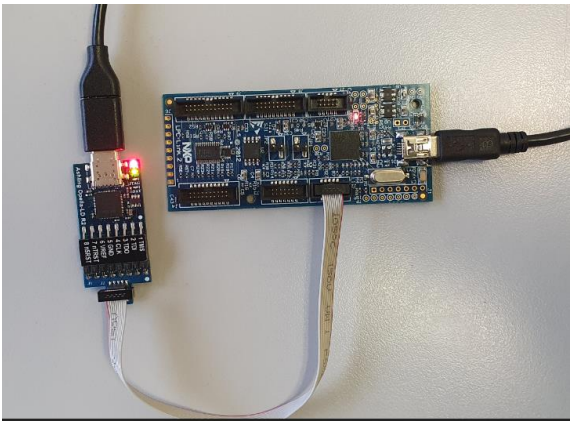

Figure 2. Opella-LD connected to an Arm target using the provided OPLD-MIPI-10 adapter

1.2 Obtaining the latest Opella-LD Documentation and Software

The latest versions of documentation and required Opella-LD software is always available on the Ashling Opella-LD support Web site at:

https://www.ashling.com/Opella-LD/

Download and unzip the software (maintaining the directory structure) before proceeding to the next stage.

1.3 Connecting the Opella-LD to your PC and Target

Opella-LD is designed to connect to your PC via the USB Port and your target as outlined in **Appendix A. Debug Connections**. Please note the following recommended target connection sequence:

1. Ensure your target is powered off.
2. Connect Opella-LD to your PC using the supplied USB cable and ensure Opella-LD's **PWR** LED is on.
3. For Windows users:
    a. You will get a **New USB hardware found** message from Windows and will be prompted to install the appropriate USB drivers. The Ashling Opella-LD driver is located in:
    `Opella-LD\Opella-LD_Windows_Driver\`
    b. Point the Windows **Hardware Installation Wizard** to your download directory so that it can locate the necessary drivers and complete the installation. Windows only needs to perform this operation the first time you connect your Opella-LD to the PC.
4. For Linux users:
    a. To ensure the current `$USER` has access to the Opella-LD device we recommend using the Linux utility `udev`.
    b. Create a `udev` rules file to uniquely identify the Opella-LD device and set permissions as required by owner/ groups. An example `udev` file provided:
    `Opella-LD\ Opella-LD_Linux_Rules_File\60-ashling.rules`
    which identifies Opella-LD device (by Ashling's USB product ID and Vendor ID).
    c. The rules file must then be copied into the rules directory (requires root permission) e.g.:
    `$ sudo cp ./60-ashling.rules /etc/udev/rules.d`
5. Connect Opella-LD to your target's debug connector (see **Appendix A. Debug Connections** for the provided Debug Connections)
6. Power up your target.

Ashling recommend you always adhere to this sequence to avoid damage to the Opella-LD or to your target. When disconnecting from your target ensure you:

1. Power off your target.
2. Power off Opella-LD (disconnect USB cable).
3. Disconnect Opella-LD from your target.

Opella-LD has four LEDs that provide useful diagnostic information, see **Appendix** B. Opella-LD LEDs for more details.

Chapter 2. Using Opella-LD with OpenOCD and GDB

2.1 Introduction

Opella-LD is a **USB FT2232 Based** Debug Adapter Hardware in OpenOCD terms. See here: http://openocd.org/doc/html/Debug-Adapter-Hardware.html#Debug-Adapter-Hardware and this section shows how to use OpenOCD (running on a Windows host) and Opella-LD with different target architectures.

2.2 Using Opella-LD and OpenOCD with Arm Targets

To run Opella-LD and OpenOCD together you will need an Opella-LD configuration file (available on www.ashling.com/Opella-LD) as follows depending on whether you are using Arm JTAG or SWD as the debug interface.

```
Opella-LD_Configs\Arm\ashling-opella-ld-jtag.cfg contents:
        adapter driver ftdi
        ftdi device_desc "Opella-LD Debug Probe"
        ftdi vid_pid 0x0B6B 0x0040
        ftdi tdo_sample_edge falling
        ftdi layout_init 0x0A68 0xFF7B
        ftdi channel 0
        ftdi layout_signal JTAGOE -ndata 0x0010
        ftdi layout_signal nTRST -data 0x0020
        ftdi layout_signal nSRST -data 0x0040
        ftdi layout_signal SWD_EN -data 0x0100
        ftdi layout_signal SWDIO_OE -data 0x0200
        ftdi layout_signal LED -ndata 0x0800
        transport select jtag
Opella-LD_Configs\Arm\ashling-opella-ld-swd.cfg contents:
        adapter driver  ftdi
        ftdi device_desc "Opella-LD Debug Probe"
        ftdi vid_pid 0x0B6B 0x0040
        ftdi layout_init 0x0860 0x0b7b
        ftdi channel 0
        ftdi layout_signal JTAGOE -data 0x0010
        ftdi layout_signal nTRST -data 0x0020
        ftdi layout_signal nSRST -data 0x0040
        ftdi layout_signal SWD_EN -data 0x0100
        ftdi layout_signal SWDIO_OE -data 0x0200
        ftdi layout_signal LED -ndata 0x0800
        transport select swd
```

The vital information above in red tells OpenOCD what the USB PID and VID of the Opella-LD Debug Probe is which allows OpenOCD to “find” and communicate with Opella-LD.

In the following example, we will use an NXP LPC-Link2 board: https://www.nxp.com/design/microcontrollers-developer-resources/lpc-microcontroller-utilities/lpc-link2:OM13054 as our target system and the GDB debugger will connect to Opella-LD via OpenOCD and download a program.

OpenOCD and the GDB debugger software is not supplied by Ashling, and it is assumed you have already installed this. For example:

- OpenOCD can be downloaded from here: https://github.com/xpack-dev-tools/openocd-xpack/releases/ and there are more install details here: https://xpack.github.io/openocd/install/

- GNU Arm Embedded GCC and be downloaded from here: https://github.com/xpack-dev-tools/arm-none-eabi-gcc-xpack/releases/ and there are more install details here: https://xpack.github.io/arm-none-eabi-gcc/install/

Run OpenOCD (adapt the command line to suit your OpenOCD install) from a command prompt as follows:

```
>bin\openocd.exe -f interface\ashling-opella-ld-jtag.cfg -f
board\nxp_lpc-link2.cfg
Open On-Chip Debugger 0.10.0+dev-00068-ge1e63ef30 (2020-01-13-17:39)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
jtag
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
Info : clock speed 25000 kHz
Info : JTAG tap: lpc4370.m4 tap/device found: 0x4ba00477 (mfg: 0x23b
(ARM Ltd.), part: 0xba00, ver: 0x4)
Info : JTAG tap: lpc4370.m0app tap/device found: 0x0ba01477 (mfg:
0x23b (ARM Ltd.), part: 0xba01, ver: 0x0)
Info : JTAG tap: lpc4370.m0sub tap/device found: 0x0ba01477 (mfg:
0x23b (ARM Ltd.), part: 0xba01, ver: 0x0)
Info : lpc4370.m4: hardware has 6 breakpoints, 4 watchpoints
Info : DAP transaction stalled (WAIT) - slowing down
Info : DAP transaction stalled (WAIT) - slowing down
Info : DAP transaction stalled (WAIT) - slowing down
Info : DAP transaction stalled (WAIT) - slowing down
Info : DAP transaction stalled (WAIT) - slowing down
Info : DAP transaction stalled (WAIT) - slowing down
Info : DAP transaction stalled (WAIT) - slowing down
Info : DAP transaction stalled (WAIT) - slowing down
Info : lpc4370.m0app: hardware has 2 breakpoints, 1 watchpoints
```

```
Info : lpc4370.m0sub: hardware has 2 breakpoints, 1 watchpoints
Info : lpc4370.m0app: external reset detected
Info : lpc4370.m0sub: external reset detected
Info : Listening on port 3333 for gdb connections
Info : Listening on port 3334 for gdb connections
Info : Listening on port 3335 for gdb connections
```

and run GDB (adapt the command line to suit your GDB install) as follows:

```
>gdb\arm-none-eabi-gdb.exe
gdb\arm-none-eabi-gdb.exe: warning: Couldn't determine a path for
the index cache directory.
GNU gdb (GNU Arm Embedded Toolchain 9-2020-q2-update)
8.3.1.20191211-git
Copyright (C) 2019 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --target=arm-
none-eabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) file lpc_link_2_spd/lpcperftest/Debug/lpcperftest.axf
Reading symbols from
lpc_link_2_spd/lpcperftest/Debug/lpcperftest.axf...
(gdb) target remote localhost:3333
Remote debugging using localhost:3333
0x10401fb0 in ?? ()
(gdb) load
Loading section .text, size 0x424 lma 0x10000000
Loading section .data, size 0x10000 lma 0x10000424
Start address 0x10000300, load size 66596
Transfer rate: 528 KB/sec, 11099 bytes/write.
```

As you can see above, GDB runs, connects to OpenOCD and downloads a program to the target (via OpenOCD and Opella-LD).

2.3 Using Opella-LD and OpenOCD with Synopsys ARC Targets

Synopsys provide a GNU Toolchain for DesignWare ARC Processors and this section shows how to debug using the Synopsys provided Eclipse CDT Debugger or the command-line GDB Debugger along with OpenOCD and Opella-LD on the Synopsys EMSK board.

Note: the Synopsys MetaWare Development Toolkit (MWDT) including the ARC MDB or MIDE debuggers are not supported by Opella-LD…please use Opella-XD.

To run Opella-LD and OpenOCD together you will need an Opella-LD configuration file (available on www.ashling.com/Opella-LD) as follows depending on whether you are using ARC JTAG or cJTAG as the debug interface.:

```
Opella-LD_Configs\ARC\ashling-opella-ld-arc-jtag.cfg contents:
        interface ftdi
        ftdi_device_desc "Opella-LD Debug Probe"
        ftdi_vid_pid 0x0B6B 0x0040
        #ftdi_tdo_sample_edge falling
        ftdi_layout_init 0x0A68 0xFF7B
        ftdi_channel 0
        ftdi_layout_signal JTAGOE -ndata 0x0010
        ftdi_layout_signal nTRST -data 0x0020
        ftdi_layout_signal nSRST -data 0x0040
        ftdi_layout_signal SWD_EN -data 0x0100
        ftdi_layout_signal SWDIO_OE -data 0x0200
        ftdi_layout_signal LED -ndata 0x0800
        transport select jtag

Opella-LD_Configs\ARC\ashling-opella-ld-arc-cjtag.cfg contents:
        interface ftdi
        ftdi_device_desc "Opella-LD Debug Probe"
        ftdi_vid_pid 0x0B6B 0x0040
        #ftdi_tdo_sample_edge falling
        ftdi_layout_init 0x0A68 0xFF7B
        ftdi_channel 0
        ftdi_layout_signal JTAGOE -ndata 0x0010
        ftdi_layout_signal nTRST -data 0x0020
        ftdi_layout_signal nSRST -data 0x0040
        ftdi_layout_signal SWD_EN -data 0x0100
        ftdi_layout_signal SWDIO_OE -data 0x0200
        ftdi_layout_signal LED -ndata 0x0800
        transport select cjtag
```

The vital information above in red tells OpenOCD what the USB PID and VID of the Opella-LD Debug Probe is which allows OpenOCD to “find” and communicate with Opella-LD.

2.3.1 ARC Debugging via JTAG

1. Ensure you have installed the Synopsys GNU Toolchain and this section assumes it is residing in the default `C:\arc_gnu\` directory.

2. Copy the `ashling-opella-ld-arc-jtag.cfg` file to `C:\arc_gnu\share\openocd\scripts\interface\ftdi\`

3. Edit the Synopsys supplied file:
   `C:\arc_gnu\share\openocd\scripts\board\snps_em_sk.cfg` and change the following line as shown in bold:

   ```
   source [find interface/ftdi/ashling-opella-ld-arc-jtag.cfg]
   ```

4. Start OpenOCD as follows:

```
C:\arc_gnu\share\openocd\scripts>C:\arc_gnu\bin\openocd.exe -f
C:\arc_gnu\share\openocd\scripts\board\snps_em_sk.cfg
Open On-Chip Debugger 0.9.0-dev-g842d4db-dirty (2020-10-20-
19:23)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.sourceforge.net/doc/doxygen/bugs.html
adapter speed: 5000 kHz
Warn : Transport "jtag" was already selected
Info : clock speed 5000 kHz
Info : JTAG tap: arc-em.cpu tap/device found: 0x200044b1 (mfg:
0x258, part: 0x0004, ver: 0x2)
Info : JTAG tap: arc-em.cpu tap/device found: 0x200044b1 (mfg:
0x258, part: 0x0004, ver: 0x2)
target state: halted
Info : accepting 'gdb' connection on tcp/3333
```

You can now start debugging using GDB or Eclipse CDT

5. GDB Debugging

```
C:\arc_gnu\bin>arc-elf32-gdb
D:\Work_Related\Workspaces\ARC_GNU_IDE_Workspace\emsk_sum\Debug\
emsk_sum.elf
C:\arc_gnu\bin\arc-elf32-gdb.exe: warning: Couldn't determine a
path for the index cache directory.
GNU gdb (ARCompact/ARCv2 ISA elf32 toolchain 2020.09)
10.0.50.20200611-git
Copyright (C) 2020 Free Software Foundation, Inc.
```

```
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute
it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=i686-w64-mingw32 --
target=arc-elf32".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://github.com/foss-for-synopsys-dwc-arc-
processors/toolchain/issues>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from
D:\Work_Related\Workspaces\ARC_GNU_IDE_Workspace\emsk_sum\Debug\
emsk_sum.elf...
(gdb) target remote :3333
Remote debugging using :3333
0x00000004 in ?? ()
(gdb) load
Loading section .init, size 0x22 lma 0x100
Loading section .text, size 0x1648 lma 0x124
Loading section .fini, size 0x16 lma 0x176c
Loading section .rodata, size 0x4 lma 0x1784
Loading section .ivt, size 0x54 lma 0x1788
Loading section .data, size 0x82c lma 0x37dc
Loading section .ctors, size 0x8 lma 0x4008
Loading section .dtors, size 0x8 lma 0x4010
Loading section .sdata, size 0x10 lma 0x4018
Start address 0x00000124, load size 7972
Transfer rate: 210 KB/sec, 885 bytes/write.
(gdb)
```

6. If you want to use Eclipse-CDT, then select **JTAG via OpenOCD** as follows in the Eclipse-CDT Debug Launch:

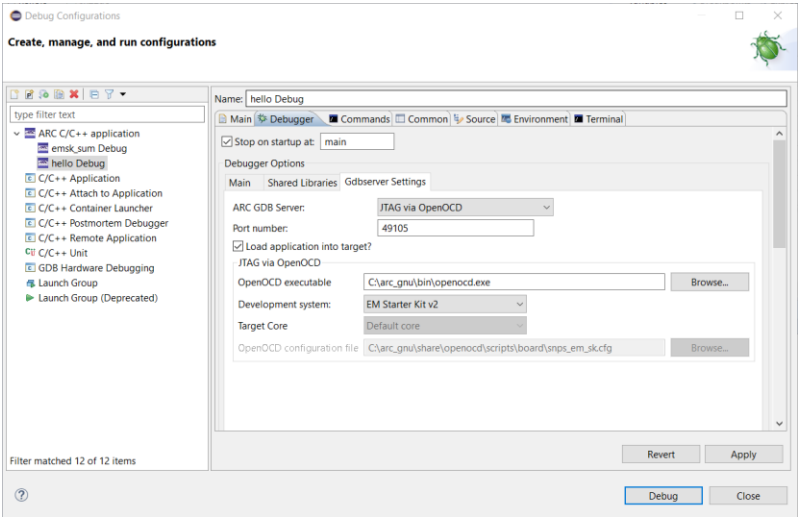

Figure 3. Eclipse-CDT Debug Launch for OpenOCD/Opella-LD

2.3.2 ARC Debugging via cJTAG

To use the Opella-LD with OpenOCD for cJTAG targets requires a modified OpenOCD. You can use the Ashling supplied binary (`Opella-LD\OpenOCD_Binaries\<Host_OS>\ARC\openocd`) or rebuild OpenOCD using the Ashling supplied patch (`Opella-LD\OpenOCD_Patches\arc_openocd_opella-ld_cjtag_0.9.0.patch`).

The steps for applying the patch and building are as follows:

1. Clone OpenOCD source code from https://github.com/foss-for-synopsys-dwc-arc-processors/openocd
2. Apply the Ashling patch from the "`src`" directory of the source code.
   ```
   > patch -p0 < arc-openocd-opella-ld-cjtag-XXX.patch
   ```
3. Build the source code.

Further references for building OpenOCD are available at:

- https://mindchasers.com/dev/openocd-darsena-windows
- https://elinux.org/Compiling_OpenOCD

Debugging for cJTAG is similar to JTAG except that you need to use the cJTAG configuration file (`ashling-opella-ld-arc-cjtag.cfg`) when running OpenOCD.

2.4 Using Opella-LD and OpenOCD with RISC-V Targets

To run Opella-LD and OpenOCD together you will need an Opella-LD configuration file (available on www.ashling.com/Opella-LD) as follows depending on whether you are using RISC-V JTAG or cJTAG as the debug interface.

```
Opella-LD_Configs\RISC-V\ashling-opella-ld-riscv-jtag.cfg contents:
        adapter driver ftdi
        ftdi_device_desc "Opella-LD Debug Probe"
        ftdi_vid_pid 0x0B6B 0x0040
        ftdi_tdo_sample_edge falling
        ftdi_layout_init 0x0A68 0xFF7B
        ftdi_channel 0
        ftdi_layout_signal JTAGOE -ndata 0x0010
        ftdi_layout_signal nTRST -data 0x0020
        ftdi_layout_signal nSRST -data 0x0040
        ftdi_layout_signal SWD_EN -data 0x0100
        ftdi_layout_signal SWDIO_OE -data 0x0200
        ftdi_layout_signal LED -ndata 0x0800
        transport select jtag

Opella-LD_Configs\RISC-V\ashling-opella-ld-riscv-cjtag.cfg contents:
        adapter driver ftdi
        ftdi_device_desc "Opella-LD Debug Probe"
        ftdi_vid_pid 0x0B6B 0x0040
        ftdi_tdo_sample_edge falling
        ftdi_layout_init 0x0A68 0xFF7B
        ftdi_channel 0
        ftdi_layout_signal JTAGOE -ndata 0x0010
        ftdi_layout_signal nTRST -data 0x0020
        ftdi_layout_signal nSRST -data 0x0040
        ftdi_layout_signal SWD_EN -data 0x0100
        ftdi_layout_signal SWDIO_OE -data 0x0200
        ftdi_layout_signal LED -ndata 0x0800
        ftdi_layout_signal TCK -data 0x0001
        ftdi_layout_signal TDI -data 0x0002
        ftdi_layout_signal TDO -input 0x0004
        ftdi_layout_signal TMS -data 0x0008
        transport select cjtag
```

The vital information above in red tells OpenOCD what the USB PID and VID of the Opella-LD Debug Probe is which allows OpenOCD to “find” and communicate with Opella-LD.

RISC-V OpenHW CORE-V users will find a specific section (**2.4.3**) which provides details on using Opella-LD with their CORE-V device.

2.4.1 RISC-V Debugging via JTAG

In the following example, we will use a Digilent Arty 100T board: https://reference.digilentinc.com/programmable-logic/arty/start as our target system and the GDB debugger will connect to Opella-LD via OpenOCD and download a program.

OpenOCD and the GDB debugger software is not supplied by Ashling, and it is assumed you have already installed this. For example:

- OpenOCD can be downloaded from here: https://github.com/sifive/freedom-tools/releases
- The toolchain can be downloaded from here: https://www.embecosm.com/resources/tool-chain-downloads/

Run OpenOCD (adapt the command line to suit your OpenOCD install) from a command prompt as follows:

```
>bin\openocd.exe -f interface\ashling-opella-ld-riscv-jtag.cfg -f
arty-100T.cfg
Open On-Chip Debugger 0.11.0+dev-01744-gb36d17672-dirty (2021-07-06-
19:56)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
transport is jtag
Info : clock speed 1000 kHz
Info : JTAG tap: riscv.cpu tap/device found: 0x20000913 (mfg: 0x489
(SiFive Inc), part: 0x0000, ver: 0x2)
Info : datacount=1 progbufsize=16
Info : Disabling abstract command reads from CSRs.
Info : Examined RISC-V core; found 1 harts
Info :  hart 0: XLEN=32, misa=0x40901105
Info : starting gdb server for riscv.cpu.0 on 3333
Info : Listening on port 3333 for gdb connections
Info : Found flash device 'sp s25fl128s' (ID 0x00182001)
Ready for Remote Connections
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
```

and run GDB (adapt the command line to suit your GDB install) as follows:

```
>gdb\riscv64-unknown-elf\bin>riscv64-unknown-elf-gdb.exe
GNU gdb (GDB) 11.0.50.20210316-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
```

```
This GDB was configured as "--host=x86_64-w64-mingw32 --
target=riscv64-unknown-elf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb) file arty/example/Debug/artytest.axf
Reading symbols from arty/example/Debug/artytest.axf...
(gdb) target remote localhost:3333
Remote debugging using localhost:3333
0x00000000 in ?? ()
(gdb) load
Loading section .init, size 0x70 lma 0x80000000
Loading section .text, size 0xa1d6 lma 0x80000070
Loading section .rodata, size 0x78 lma 0x8000a248
Loading section .eh_frame, size 0x2c lma 0x8000a2c0
Loading section .data, size 0x434 lma 0x8000a2f0
Start address 0x80000000, load size 42782
Transfer rate: 26 KB/sec, 6111 bytes/write.
```

As you can see above, GDB runs, connects to OpenOCD and downloads a program to the target (via OpenOCD and Opella-LD).

2.4.2 RISC-V Debugging via cJTAG

To use the Opella-LD with OpenOCD for cJTAG targets requires a modified OpenOCD. You can use the Ashling supplied binary (`Opella-LD\OpenOCD_Binaries\<Host_OS>\RISC-V\openocd`) or rebuild OpenOCD using the Ashling supplied patch (`Opella-LD\OpenOCD_Patches\riscv-openocd-opella-ld-cjtag_0_11_0.patch`).

The steps for applying the patch and building are as follows:

1. Clone OpenOCD source code from https://github.com/riscv/riscv-openocd
2. Apply the Ashling patch from the "`src`" directory of the source code.
   ```
   > patch -p0 < riscv-openocd-opella-ld-cjtag_0_11_0.patch
   ```
3. Build the source code.

Further references for building OpenOCD are available at:

- https://mindchasers.com/dev/openocd-darsena-windows
- https://elinux.org/Compiling_OpenOCD

Debugging for cJTAG is similar to JTAG except that you need to use the cJTAG configuration file (`ashling-opella-ld-riscv-cjtag.cfg`) when running OpenOCD.

2.4.3 OpenHW CORE-V RISC-V Debugging via JTAG

To use the Opella-LD with an OpenHW CORE-V target requires either the prebuilt OpenOCD and toolchain binaries from the OpenHW SDK or that you build them yourself from the CORE-V repo with a patch for the CORE-V devices. The OpenOCD and tool chain repository is here.

- https://github.com/pulp-platform/riscv-openocd
- https://github.com/pulp-platform/pulp-riscv-gnu-toolchain

2.4.3.1 CORE-V RISC-V Debugging via JTAG on the Genesys 2 board

In the following example, we will use a Genesys 2 board: https://reference.digilentinc.com/programmable-logic/genesys-2/start as our target system and the GDB debugger will connect to Opella-LD via OpenOCD and download a program.

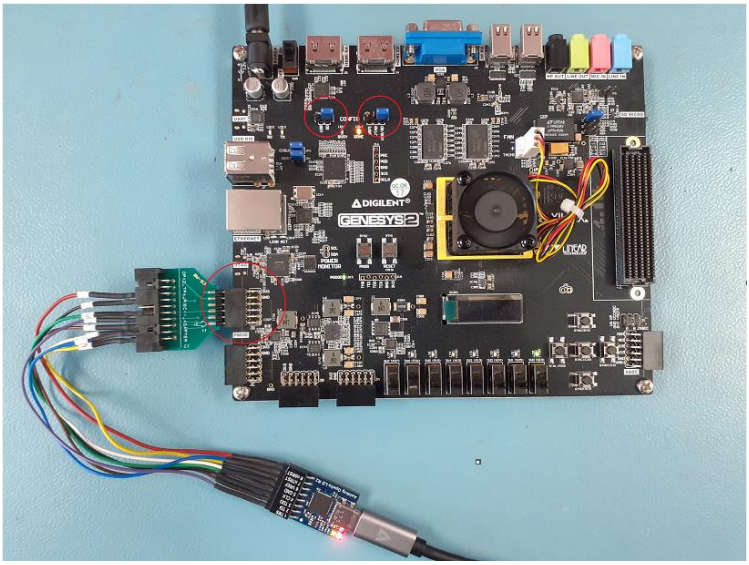

Figure 4. Genesys2 Opella-LD connection

The steps are follows:

1. The target FPGA must be pre-programmed with the corresponding CORE-V bitfile before the debugging process starts.
2. Please choose the appropriate jumper settings for programming the FPGA via USB drive, SD card or Xilinx Vivado tool. The settings shown in the above image are for programming via SD card. Please refer to the Genesys2 reference manual for more details.
3. Connect Ashling Opella-LD to the **JD** connector of the board using the flying leads as follows. Please make sure to match **Vref** of the Opella-LD and the **JD** connector. Note: The connector used and the pinout can change based on the FPGA design.

Opella-LD pin	Function	Genesys 2 JD Pin
1 TMS	JTAG Test Mode Select (TMSC in cJTAG mode or SWDIO in ARM SWD mode)	8
2 TDI	JTAG Test Data In	7
3 TDO	JTAG Test Data Out (SWO in ARM SWD mode)	1
4 CLK	JTAG/SWDCLK/cJTAG CLocK	3
5 GND	Ground	5
6 VREF	Target reference voltage used by the Opella-LD to sense target voltage (0.9v to 5.0v) and adjust probe voltages accordingly. TGT LED (YELLOW) is on when voltage detected	6
7 nTRST	Active low JTAG TAP ReSeT. Can be controlled via OpenOCD software. See http://openocd.org/doc/html/Reset-Configuration.html	No connect
8 nSRST	Active low System ReSeT. Can be controlled via OpenOCD software. See http://openocd.org/doc/html/Reset-Configuration.html	No connect

Table 1: Opella-LD Target Interface to Genesys 2 JD Connector

4. Copy the Ashling CORE-V Opella-LD config file `Opella-LD_Configs\RISC-V\ashling-opella-ld-corev-genesys2-jtag.cfg` (available on www.ashling.com/Opella-LD) to the `tcl/target` directory of OpenOCD.
5. Run OpenOCD from a command prompt as follows:

```
>bin/openocd.exe -f target/ashling-opella-ld-corev-genesys2-jtag.cfg
Open On-Chip Debugger 0.10.0+dev-00831-g348d97c58 (2021-07-19-21:25)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
Info : clock speed 1000 kHz
Info : JTAG tap: riscv.unknown0 tap/device found: 0x10102001 (mfg: 0x000 (<invalid>), part: 0x0102, ver: 0x1)
Info : JTAG tap: riscv.cpu tap/device found: 0x249511c3 (mfg: 0x0e1 (Wintec Industries), part: 0x4951, ver: 0x2)
Info : datacount=2 progbufsize=8
Info : Examined RISC-V core; found 1024 harts
Info :  hart 0: currently disabled
Info :  hart 1: currently disabled
Info :  hart 2: currently disabled
Info :  hart 3: currently disabled
Info :  hart 4: currently disabled
Info :  hart 5: currently disabled
Info :  hart 6: currently disabled
Info :  hart 7: currently disabled
```

6. Now start GBD and connect to the target and debug

```
bin>riscv32-unknown-elf-gdb.exe
GNU gdb (GDB) 11.0.50.20210316-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=x86_64-w64-mingw32 --target=riscv64-unknown-elf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
(gdb) file ../pulpissimo_sum\Debug\pulpissimo_sum.elf
Reading symbols from ..\pulpissimo_sum\Debug\pulpissimo_sum.elf...
(gdb) target remote:3333
Remote debugging using :3333
```

```
main () at ../src/sum.c:23
23              x++; y++;
(gdb) load
Loading section .text.init, size 0xb8 lma 0x1c000000
Loading section .text, size 0xa8 lma 0x1c0000b8
Start address 0x1c000000, load size 352
Transfer rate: 42 KB/sec, 176 bytes/write.
(gdb) b main
Breakpoint 1 at 0x1c000128: file ../src/sum.c, line 19.
(gdb) c
Continuing.

Breakpoint 1, main () at ../src/sum.c:19
19              int x=3,y=9;
(gdb)
```

As you can see above, GDB runs, connects to OpenOCD and downloads a program to the target (via OpenOCD and Opella-LD).

2.4.3.2 CORE-V RISC-V Debugging via JTAG on the Nexys A7 board

In this section, the steps to be followed for debugging the Digilent Nexys A7 https://digilent.com/reference/programmable-logic/nexys-a7/start target are explained.

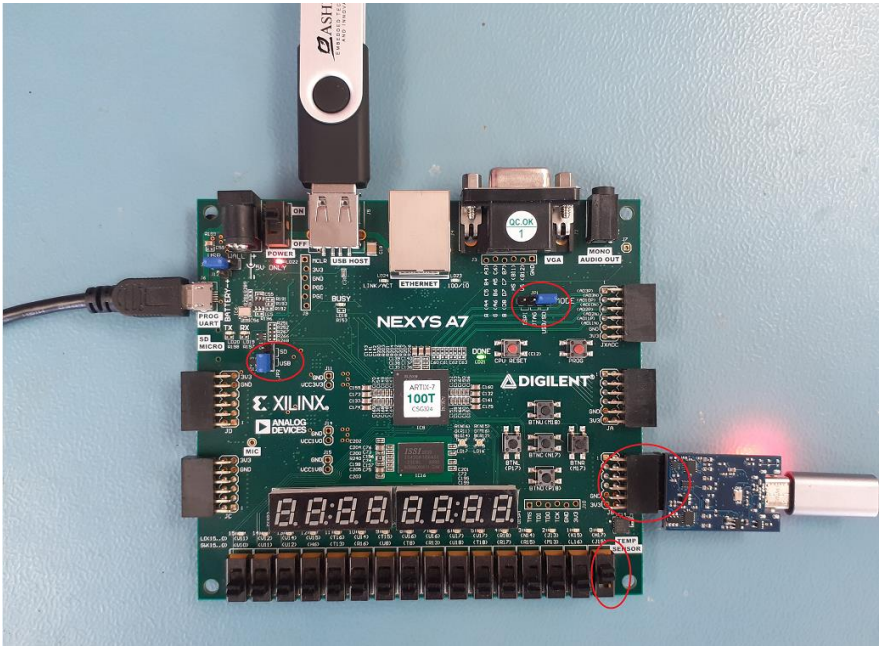

Figure 5. Nexys A7 Opella-LD connection

The steps are follows:

1. The Nexys A7 must be pre-programmed with the corresponding CORE-V cores before the debugging process starts. It can be done via a USB drive, SD card or the Xilinx Vivado tool and the jumper settings shown in the above image are for programming via USB drive. Please refer to the Nexys A7 reference manual for more information.
2. Connect Opella-LD to the bottom 6-pins of the **JB** connector. A 6-pin gender changer or flying leads can be used for this. Please make sure to match **Vref** of the Opella-LD and the **JB** connector. Note: The connector used and the pinout can change based on the FPGA design.

Opella-LD pin	Function	Nexys A7 JB pin
1 TMS	JTAG Test Mode Select (TMSC in cJTAG mode or SWDIO in ARM SWD mode)	7
2 TDI	JTAG Test Data In	8
3 TDO	JTAG Test Data Out (SWO in ARM SWD mode)	9
4 CLK	JTAG/SWDCLK/cJTAG CLocK	10
5 GND	Ground	11
6 VREF	Target reference voltage used by the Opella-LD to sense target voltage (0.9v to 5.0v) and adjust probe voltages accordingly. TGT LED (YELLOW) is on when voltage detected	12
7 nTRST	Active low JTAG TAP ReSeT. Can be controlled via OpenOCD software. See http://openocd.org/doc/html/Reset-Configuration.html	No connect
8 nSRST	Active low System ReSeT. Can be controlled via OpenOCD software. See http://openocd.org/doc/html/Reset-Configuration.html	No connect

Table 2: Opella-LD Target Interface to Nexys A7 JB Connector

3. Make sure the **SW0** switch is on (up position) to route the JTAG signals to the **JB** connector.
4. Copy the Ashling CORE-V Opella-LD config file `Opella-LD_Configs\RISC-V\ashling-opella-ld-corev-nexysa7-jtag.cfg` (available on www.ashling.com/Opella-LD) to the `tcl/target` directory of OpenOCD.
5. Run OpenOCD from a command prompt as follows:

```
>bin/openocd.exe  -f  target/ashling-opella-ld-corev-nexysa7-jtag.cfg
Open On-Chip Debugger 0.10.0+dev-00831-g348d97c58 (2021-07-19-21:25)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
Info : clock speed 1000 kHz
Info : JTAG tap: riscv.cpu tap/device found: 0x10001c05 (mfg: 0x602 (<unknown>), part: 0x0001, ver: 0x1)
Info : datacount=2 progbufsize=8
Info : Examined RISC-V core; found 1 harts
Info :  hart 0: XLEN=32, misa=0x40001104
Info : Listening on port 3333 for gdb connections
Ready for Remote Connections
Info : Listening on port 6666 for tcl connections
Info : Listening on port 4444 for telnet connections
```

6. Now start GBD and connect to the target and debug

```
bin>riscv32-unknown-elf-gdb.exe
GNU gdb (GDB) 11.0.50.20210316-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and
redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "--host=x86_64-w64-mingw32 --
target=riscv64-unknown-elf".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online
at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to
"word"...
(gdb) file ../pulpissimo_sum\Debug\pulpissimo_sum.elf
Reading symbols from
..\pulpissimo_sum\Debug\pulpissimo_sum.elf...
Remote debugging using :3333
0x1c009262 in ?? ()
(gdb) load
Loading section .text.init, size 0xb8 lma 0x1c000000
Loading section .text, size 0xa8 lma 0x1c0000b8
Start address 0x1c000000, load size 352
Transfer rate: 42 KB/sec, 176 bytes/write.
(gdb) b main
Breakpoint 1 at 0x1c000128: file ../src/sum.c, line 19.
(gdb) c
Continuing.

Breakpoint 1, main () at ../src/sum.c:19
19              int x=3,y=9;
(gdb) n
22              sum(x,y);
(gdb) s
sum (start=3, end=9) at ../src/sum.c:7
7               int  sum = 0;
(gdb)
```

As you can see above, GDB runs, connects to OpenOCD and downloads a program to the target (via OpenOCD and Opella-LD).

Chapter 3. Appendices

3.1 Appendix A. Debug Connections

Opella-LD has eight IO signals which are clearly labelled, and these provide an interface to your target system. 0.9v to 5.0v targets are supported and the Opella-LD senses and adapts automatically to your target voltage. Debug Adapters are also available to mate with existing target sockets.

Pin	IO Signal	Function	Direction
1	TMS	JTAG Test Mode Select (TMSC in cJTAG mode or SWDIO in ARM SWD mode)	Target to Opella-LD
2	TDI	JTAG Test Data In	Target to Opella-LD
3	TDO	JTAG Test Data Out (SWO in ARM SWD mode)	Target to Opella-LD
4	CLK	JTAG/SWDCLK/cJTAG CLocK	From Opella-LD to target
5	GND	Ground	-
6	VREF	Target reference voltage used by the Opella-LD to sense target voltage (0.9v to 5.0v) and adjust probe voltages accordingly. TGT LED (YELLOW) is on when voltage detected	Target to Opella-LD
7	nTRST	Active low JTAG TAP ReSeT. Can be controlled via OpenOCD software. See http://openocd.org/doc/html/Reset-Configuration.html	Opella-LD to target
8	nSRST	Active low System ReSeT. Can be controlled via OpenOCD software. See http://openocd.org/doc/html/Reset-Configuration.html	Opella-LD to target

Table 3: Opella-LD Target Interface

3.1.1 Available Debug Adapters

3.1.1.1 OPLD-MIPI-10

The OPLD-MIPI-10 adapter supports the MIPI and Arm CoreSight 10-way 0.05” target sockets

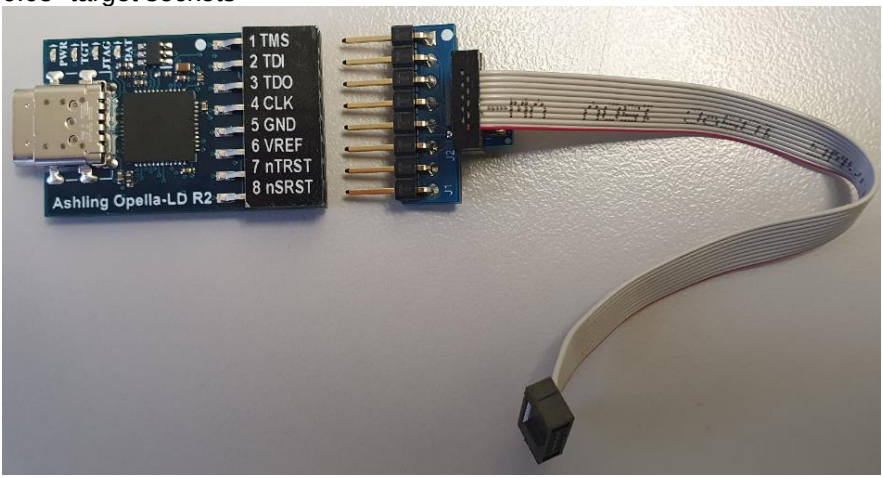

Figure 6. The Opella-LD OPLD-MIPI-10 adapter (on the right) provides support for targets using a 10-way 0.05” socket (e.g. MIPI or Arm CoreSight)

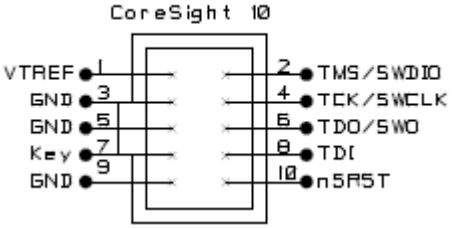

Figure 7. OPLD-MIPI-10 Coresight 10-way 0.05” socket Pinout

3.1.1.2 Flying Leads

The Opella-LD Flying Leads allow direct connection to 0.1” pins on your target. Each lead is numbered from 1 to 8. Ensure you connect GND (5) and VREF (6) as well as the necessary data signals TMS (1), TDI (2), TDO (3) and CLK (4).

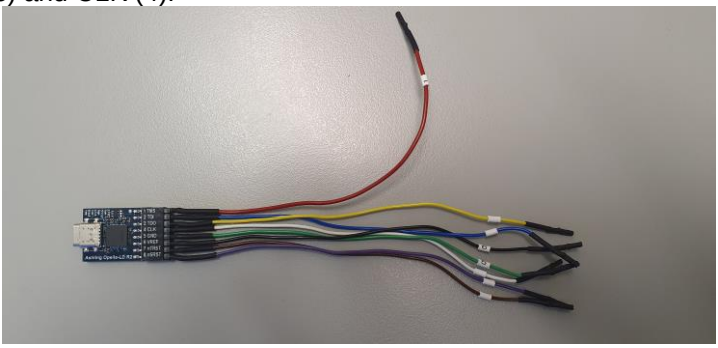

Figure 8. The Opella-LD Flying Leads

3.1.1.3 Blanking Pins (x2)

Some targets (e.g. Synopsys ARC) only use a 6-pin connection and the blanking pins allow you to block of pins 7 and 8 to prevent incorrect connection.

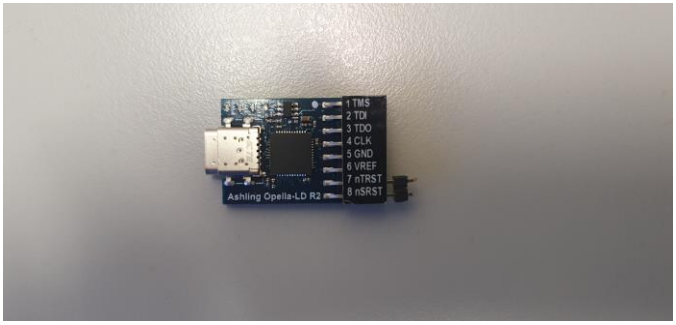

Figure 9. The Opella-LD Blanking Pins (pins 7 and 8)

3.2 Appendix B. Opella-LD LEDs

The following table describes the LEDs on the Opella-LD:

LED	State	Meaning
PWR	RED	Opella-LD powered and connected to PC via USB
TGT	YELLOW	Target detected (0.9v to 5.0v)
JTAG	BLUE	On indicates JTAG Data Transfer Mode (off indicates SWD Data Transfer Mode)
DAT	GREEN	Indicates Data traffic between Opella-LD and Target

Table 4. Opella-LD LEDs