

Readme - Simplified CDI2 Device

Peter Schurek, Benjamin Venditti
Austrian Institute of Technology GmbH, Giefinggasse 4 - 1210 Vienna - Austria

September 12, 2023

Contents

CDI2 Device Interface Xilinx FPGA Design	1
1. FPGA Design	1
2. Embedded SW	1
2.1 Update FPGA Bitstream with ELF	2
2.2 Debugging	2
3. PetaLinux	3
3.1 SD Card Boot	3
3.2 QSPI Boot	3
4. Running the TestApp	4
4.1 Connecting the device to the BSMS	4
4.2 TestApp	4
4.3 Required services	6
5. Bugs	6

CDI2 Device Interface Xilinx FPGA Design

This project comes with all external IP cores not included in a standard Vivado installation and the TCL script generating the synthesizable block design.

1. FPGA Design

In order to setup the project:

- start Vivado
- on the TCL command shell, change into the design's root directory (e.g., FPGA/Xilinx) and source the project nucleus script

```
cd ./Public/Project/FPGA/Xilinx
source ./scripts/recreate_Mars_ZX2_ST3.tcl
```

To update the nucleus script execute the `write_project_tcl` command

```
cd [get_property DIRECTORY [current_project]]
cd ..
write_project_tcl -force ./recreate.tcl
```

, then move the TCL script into the `scripts` folder

```
mv recreate.tcl scripts/recreate_Mars_ZX2_ST3.tcl
```

2. Embedded SW

The `Workspace` directory contains the BSP, OpenPowerLink stack, and CDI2 application projects with all sources. It is an export of the actual Vitis/eclipse project and can be used to re-create the embedded

SW for the CDI2 Device core. However, the original ZIP archive is too coarse granular for source control and QM purposes and furthermore hangs the IDE when naively imported.

- Copy the contents of the directory to your final Workspace location.
- Start the Xilinx Vitis IDE and point its Workspace to your chosen location
- Do not generate any BSP, but just import the existing Workspace, but make sure uncheck the 'copy into workspace' checkbox

2.1 Update FPGA Bitstream with ELF

It is possible to update the BRAM initialization in the FPGA bitstream with a newer CDI2 Device Core SW without executing a full Vivado Synthesis. The exported HW archive, Mars_ZX2_ST3.xsa, contains all the necessary data. Conveniently, its expanded version can be found in the Workspace directory Workspace/Mars_ZX2_ST3/hw or in CDI2_DeviceApp/project-spec/hw-description of the PetaLinux project after importing the HW description. You need bitstream, ELF, and the MMI file:

```
$ /opt/Xilinx/Vivado/2022.2/bin/updatemem --meminfo Mars_ZX2_ST3.mmi \
--data ../../CDI2_Device/Release/CDI2_Device.elf \
--bit Mars_ZX2_ST3.bit \
--proc cdi2Device/CDI2_Device_i/SoftCore \
--out test.bit
```

See the petalinux-package command below to pack this new bitstream into a new BOOT.BIN for the Zynq. However, be aware that petalinux takes the FPGA bitstream from CDI2_DeviceApp/project-spec/hw-description/Mars_ZX2_ST3.bit, and not from CDI2_DeviceApp/images/linux/system.bit as rationally expected.

2.2 Debugging

Debugging the Microblaze as 'Standalone Application' becomes unfeasable once the test driver running on PS/Linux comes into play. For hosted sessions, use the Attach to running CDI2_Device debug configuration and use the XSCT Console to reset and load the Microblaze processor:

```
xsct% targets
target
1  APU
2  ARM Cortex-A9 MPCore #0 (Running)
3  ARM Cortex-A9 MPCore #1 (Running)
4  xc7z020
5  MicroBlaze Debug Module at USER2
6  MicroBlaze #0 (Running)
xsct% target 6
xsct% rst
xsct% Info: MicroBlaze #0 (target 6) Stopped at 0x0 (External debug request)
xsct% dow Mars_ZX2_ST3/Workspace/CDI2_Device/Debug/CDI2_Device.elf
```

```
Downloading Program -- /home/peter/Projects/Spectra/Mars_ZX2_ST3/ \
Workspace/CDI2_Device/Debug/CDI2_Device.elf
section, .vectors.reset: 0x00000000 - 0x00000007
section, .vectors.sw_exception: 0x00000008 - 0x0000000f
section, .vectors.interrupt: 0x00000010 - 0x00000017
section, .vectors.hw_exception: 0x00000020 - 0x00000027
section, .text: 0x00000050 - 0x0004829b
[...]
80%    OMB    0.0MB/s  00:01 ETA
90%    OMB    0.0MB/s  00:00 ETA
98%    OMB    0.0MB/s  00:00 ETA
100%   OMB    0.0MB/s  00:07
```

```

Setting PC to Program Start Address 0x00000000
Successfully downloaded /home/peter/Projects/Spectra/Mars_ZX2_ST3/ \
Workspace/CDI2_Device/Debug/CDI2_Device.elf

xsct% con
Info: MicroBlaze #0 (target 6) Running
xsct% Info: MicroBlaze #0 (target 6) Stopped at 0x1c02c (Breakpoint)
main() at ../src/main.c: 566
566:      xil_printf("*** CDI2 Device Interface Core ***\r\n");
xsct%

```

3. PetaLinux

Download the PetaLinux SDK from the Xilinx supportsite and install as standard user (root is not allowed) in directory \$PETALINUX. In our case this is '\$HOME/Projects/Spectra/petalinux'. Create a project CDI2_DeviceApp.

```

$ source $PETALINUX/settings.sh
$ petalinux-create -type project -template zynq -name CDI2_DeviceApp

```

Copy the project-spec directory from our instance into your project root and build everything:

```

$ cp -a <project-spec> CDI2_DeviceApp
$ cd CDI2_DeviceApp
# update HW description, just in case
$ petalinux-config --get-hw-description <Mars_ZX2_ST3.xsa>
# build fsbl, u-boot, kernel, device tree, and root fs
$ petalinux-build
# generate a BOOT.BIN
$ cd images/linux
$ petalinux-package --boot --fsbl --fpga --u-boot --force \
    --bootgen-extra-args "-log trace"

```

3.1 SD Card Boot

Prepare an SD Card for booting: With gparted(8) create a FAT32 partition (4G) and a ext4 partition (rest). Copy BOOT.BIN, boot.scr, and image.ub to the FAT boot partition of the SD card, expand rootfs.ext.tar on the EXT4 partition.

Make sure the dip switches, next to the SD card slot, on the ST3 Board are switched correctly (1:OFF, 2:OFF, 3:ON, 4:OFF)

The serial console of the ARM processor is connected to an FTDI UART/USB bridge on the ST3 base board. Compatible settings are 8N1, 115200 baud, no HW and no SW flow control.

The standard user is `petalinux`; you are forced to set a password on first login.

3.2 QSPI Boot

BOOT.BIN can also be used to boot from QSPI. Boot PetaLinux from SD card, log in and become root:

```

$ sudo su
# mount /dev/mmcblk0p1 /mnt
# flashcp -vA /mnt/BOOT.BIN /dev/mtd0

```

In principle, the Xilinx FSBL can boot from SD card or QSPI, but not in practice: the QSPI bus width must be configured to 1 bit. This is now added as patch in the PetaLinux meta-user layer. Contrary to documentation, the FSBL is build with recipe fsbl-firmware, not virtual/bootloader:

```

$ cd CDI2_DeviceApp
$ petalinux-build -c fsbl-firmware -x cleanall
$ petalinux-build -c fsbl-firmware

```

```
$ cd images/linux
$ petalinux-package --boot --fsbl --fpga --u-boot --force \
    --bootgen-extra-args "-log trace"
```

4. Running the TestApp

To run the included TestApp the device has to be connected to the CDI2 BSM Simulator first.

4.1 Connecting the device to the BSMS

To connect the device to the BSMS as a **camera**, disconnect the PowerLink wire from device 1 (of the DS - Device Simulator) and plug it into the PL port on the phy board. Additionally disconnect the TTS cable from device 1 and plug it into the TTS connector on the phy board.

If you wish to test the device as a **detector**, please do so accordingly but with the connectors of device no. 2 (of the DS - Device Simulator).

4.2 TestApp

4.2.1 Starting the TestApp To run the TestApp you may use the following command.

```
cdi2app.sh [-r|--role arg] [-i|--nodeId arg] [-d|device arg] [-s|--resultSize arg] \
    [-l|--log arg] [--help] [--noTTS]
-r, --role          Set the cdi2 device role [Camera|Detector]
-i, --nodeId        Set the cdi2 device node id [1,16], default is 1/2 depending on \
                    the role Camera/Detector
-d, --device        Set the uart devices to use, default is '/dev/ttyS0,/dev/ttyS1'
-s, --resultSize    Set the size of the BnResult [15,1412] being sent.
-l, --log           Set the log level [DEBUG, INFO, WARNING, ERROR, CRITICAL], \
                    default is 'INFO'.
--noTTS            Disable TTS
--help             Prints this description.
```

Executed with default parameters:

```
$sudo ./cdi2app.sh -r Camera -d /dev/ttyS0,/dev/ttyS1
```

This will start the TestApp as a **camera** with powerlink node id=1 on serial port /dev/ttyS0. To start the application as a **detector** please adjust the arguments accordingly. The script will also start all required services (pppd, ftpd). Once the application has been started, you may run the testcase on the BSM simulator. Please note, that you will have to start the corresponding test case on the DS (Device Simulator) if you are testing other devices than your own.

:warning: Changing the role Camera|Detector will require a restart of the system.

The output should look something like this:

```
$ sudo ./cdi2app.sh -d /dev/ttyS0,/dev/ttyS1 -r Camera
___: 2023-09-07 12:48:18,623 INFO: Using ROLE=Camera, UART1=/dev/ttyS0, \
    UART2=/dev/ttyS1, NODE_ID=1, TTS=, SIZE=, LOG_LEVEL=INFO
WEB: 2023-09-07 12:48:18,709 INFO: Starting webserver (python http.server) in background
FTP: 2023-09-07 12:48:18,715 INFO: Starting tcpsvd (ftpd) in background
PPP: 2023-09-07 12:48:18,727 INFO: Preparing ip-up.d script
FTP: 2023-09-07 12:48:18,737 tcpsvd: listening on 0.0.0.0:21, starting
APP: 2023-09-07 12:48:18,716 INFO: Starting cdi2DeviceApp.py in background
PPP: 2023-09-07 12:48:18,747 INFO: Starting pppd in background
APP: 2023-09-07 12:48:18,771 INFO: Setting role to 'Camera' with './ftp/device_info-cs.xml'
PPP: 2023-09-07 12:48:18,771 using channel 3
PPP: 2023-09-07 12:48:18,796 Using interface ppp0
PPP: 2023-09-07 12:48:18,823 Connect: ppp0 <--> /dev/pts/1
APP: 2023-09-07 12:48:18,787 INFO: Updated device_info.xml: ttsSupport="true"
```

```

PPP: 2023-09-07 12:48:18,855 sent [LCP ConfReq id=0x1]
PPP: 2023-09-07 12:48:18,882 rcvd [LCP ConfAck id=0x1]
PPP: 2023-09-07 12:48:18,896 rcvd [LCP ConfReq id=0x0]
APP: 2023-09-07 12:48:18,874 INFO: Updated device_info.xml: <CameraSystem nodeId="1" \
    useSecondBfa="false">
PPP: 2023-09-07 12:48:19,084 sent [LCP ConfAck id=0x0]
PPP: 2023-09-07 12:48:19,126 kernel does not support PPP filtering
PPP: 2023-09-07 12:48:19,148 sent [IPCP ConfReq id=0x1 <addr 192.168.100.1>]
PPP: 2023-09-07 12:48:19,236 rcvd [IPCP ConfAck id=0x1 <addr 192.168.100.1>]
APP: 2023-09-07 12:48:19,932 INFO: TOP: connect and configure UART interface /dev/ttyS0
APP: 2023-09-07 12:48:19,946 INFO: RA: CDI2 Message Reassembly task
APP: 2023-09-07 12:48:19,947 INFO: APP: CDI2 Device Application
APP: 2023-09-07 12:48:19,948 INFO: APP: Protocol Version 2.0
APP: 2023-09-07 12:48:19,951 INFO: DEV: IP Core status 0, protocol 2.0, BSM 0, DET 10, NMT 0x1d
PPP: 2023-09-07 12:48:21,796 sent [IPCP ConfReq id=0x1 <addr 192.168.100.1>]
PPP: 2023-09-07 12:48:21,807 rcvd [IPCP ConfAck id=0x1 <addr 192.168.100.1>]
PPP: 2023-09-07 12:48:21,818 rcvd [IPCP ConfReq id=0x0 <addr 192.168.100.254>]
PPP: 2023-09-07 12:48:21,829 sent [IPCP ConfAck id=0x0 <addr 192.168.100.254>]
PPP: 2023-09-07 12:48:21,839 local IP address 192.168.100.1
PPP: 2023-09-07 12:48:21,850 remote IP address 192.168.100.254
PPP: 2023-09-07 12:48:21,859 Script /etc/ppp/ip-up started (pid 1255)
PPP: 2023-09-07 12:48:21,868 Script /etc/ppp/ip-up finished (pid 1255), status = 0x0
FTP: 2023-09-07 12:48:22,705 INFO: Checking ftpd
FTP: 2023-09-07 12:48:22,715 tcpsvd: start 1278 127.0.0.1:21-127.0.0.1:46136
FTP: 2023-09-07 12:48:22,724 tcpsvd: status 1/30
FTP: 2023-09-07 12:48:22,733 ftpd[1278]: 220 Operation successful
PPP: 2023-09-07 12:48:28,717 INFO: Checking routing table
PPP: 2023-09-07 12:48:28,729 10.111.200.0/24 dev eth0 proto kernel scope link src 10.111.200.88
PPP: 2023-09-07 12:48:28,740 192.168.100.0/24 dev ppp0 scope link
PPP: 2023-09-07 12:48:28,750 192.168.100.254 dev ppp0 proto kernel scope link src 192.168.100.1
PPP: 2023-09-07 12:48:28,759 INFO: routing table ok

```

Once the TestApp has started successful you may run the test from the BSMS.

4.2.2 BSMS Console Test In the simulation gui of the BSMS activate the testcase BSMS_BSM-G01 and connect via SSH to the BSM Node Controller (MN, 10.111.200.66) to execute testcase with the following commands:

```

$ cd /var/tmp/BSMS_BSM-G01
$ bsm-sim --test-file sorting_test.lua --config config_bsm.xml -v9 \
    --device-selection 1 --result-dir . --log-dir log

```

This will only test the device no. 1 connected to the DS.

4.2.3 BSMS Web UI Test In the web-ui of the BSMS activate and start the testcase BSMS_BSM-G01. If you only wish to test a single device, configure the test parameter BSM Command Line: --device-selection 1 of the simulation gui (web).

4.2.4 Configuration Please note, that all configuration is done automatically by the cdi2app.sh script.

If the device will be controlled via a FTDI usb to serial connector (e.g. ttyUSB0) the latency_timer needs to be adjusted with the following command:

```

sudo bash -c "echo 0 > /sys/bus/usb-serial/devices/ttyUSB0/latency_timer"

```

The following table contains an overview of the two exemplary device descriptions that are included in the ftp folder.

File	Mode	Options
device_info-cs.xml	Camera	NodeID 1 TTS enabled
device_info-det.xml	Detector	NodeID 2, TTS enabled

4.3 Required services

You do not have to start the services on your own. The script `cdi2app.sh` will start all required services for you.

4.3.1 PPPD On linux side, `pppd(8)` must be started to activate the PPP link and create the network interface `ppp0`. For everything to work properly, two configuration files are provided:

- `options.ttyS1` - the configuration of the `pppd(8)` daemon.
- `50cdi2` - A “link-up” script executed by `pppd(8)` whenever a PPP link becomes available. Must be copied into `/etc/ppp/ip-up.d` with world execute permissions. This script installs a route to the PowerLink network `192.168.100.0/24`

For full observability, `pppd(8)` can be started with options `nodetach`, `debug`, and `record`. The session log `ppp.log` can be inspected with `pppdump(8)` or analyzed with Wireshark.

```
$ sudo pppd /dev/ttyS1 1152000 192.168.100.1: file options.uart2 \
nodetach debug record ppp.log
```

To check that the pppd tunnel is working correctly, please check the routing information. The output should look something like this.

```
$ip route
10.111.200.0/24 dev eth0 proto kernel scope link src 10.111.200.50
192.168.100.0/24 dev ppp0 scope link
192.168.100.254 dev ppp0 proto kernel scope link src 192.168.100.1
```

4.3.2 FTPD For exchanging the XML configuration files, an FTP server is needed on device side. Examples of these files are provided with the application, in directory `ftp` which also may serve as root directoy for the anonymous FTP server:

```
$ cd ~petalinux/App
$ sudo tcpsvd -vE 0.0.0.0 21 ftpd -w -A -vv ./ftp
```

:warning: Please note, that the ftp server employed is a patched version of busybox ftpd. To run ftpd on an external computer, ftpd has to be patched with `0001-ftpd.patch` and compiled on the target system. You may compile a custom version of ftpd (busybox) with the shell script.

```
$ ./compile-busybox.sh
```

5. Bugs

Link Error The example project puts the standard Xilinx, C, and GCC runtime libraries into a archive group to resolve circular dependencies. Unfortunately, Vitis does not recognize the `-Wl,--start-group ... --end-group` syntax and, assuming these options to be link libraries, blindly prefixes them with `-l`, thus leading to the following error when linking `CDI2_Device.elf`:

```
[...]
/opt/Xilinx/Vitis/2022.2/gnu/microblaze/lin/x86_64-oesdk-linux/usr/bin \
/microblaze-xilinx-elf/microblaze-xilinx-elf-ld.real: \
cannot find -l-Wl,--start-group,-lxiltimer,-lxil,-lfreertos,-lgcc,-lc,--end-group
collect2.real: error: ld returned 1 exit status
make: *** [makefile:38: CDI2_Device.elf] Error 1
```

This happens whenever the build system is re-generated. For now, just edit the files `Workspace/CDI2_Device/Debug/objects.mk` and `Workspace/CDI2_Device/Release/objects.mk` until the list of link libraries is settled.

cmake(1) Error Vitis comes with its own `cmake(1)` installation in `/opt/Xilinx/Vitis/2022.2/tps/lnx64/cmake-3.21.4/bin`.

Unfortunately, this `cmake` is linked against `libcrypto.so.10` and `libssl.so.10` which are not part of the stock Ubuntu installation. To make `cmake(1)` work in the Vitis IDE, create symbolic links to these shared libraries in `/opt/Xilinx/Vitis/2022.2/lib/lnx64.o/Ubuntu`:

```
libcrypto.so.10 -> ../../../../tps/lnx64/cmake-3.21.4/libs/Ubuntu/libcrypto.so.10
libssl.so.10 -> ../../../../tps/lnx64/cmake-3.21.4/libs/Ubuntu/libssl.so.10
```