Zynq GEM Reference Designs

Jeff Johnson

May 18, 2022

USER GUIDE

This is the documentation for the Zynq GEM reference designs for the Ethernet FMC.

CHAPTER

ONE

DESCRIPTION

The reference designs target both Zynq boards and Zynq MP (Zynq Ultrascale+) boards. The designs for the two target devices differ slightly due to the fact that the Zynq devices have only 2x GEMs whereas the ZynqMP devices have 4x.

1.1 Zynq Designs

In the Zynq designs, the first three ports of the Ethernet FMC are connected to the AXI Ethernet Subsystem IP which are then connected to the system memory via AXI DMA IP. The fourth port of the Ethernet FMC is connected to the GMII-to-RGMII IP which connects to hard GEM0 of the Zynq PS via the FPGA fabric (EMIO). The Ethernet port of the Zynq development board (ie. the on-board Ethernet port) connects directly to the hard GEM1 of the Zynq PS via MIO.

1.2 ZynqMP Designs

In the ZynqMP designs, all ports of the Ethernet FMC are connected to the GMII-to-RGMII IP which connects to hard GEMs of the ZynqMP PS via the FPGA fabric (EMIO). The on-board Ethernet port is left unconnected and is thus unusable in these reference designs.

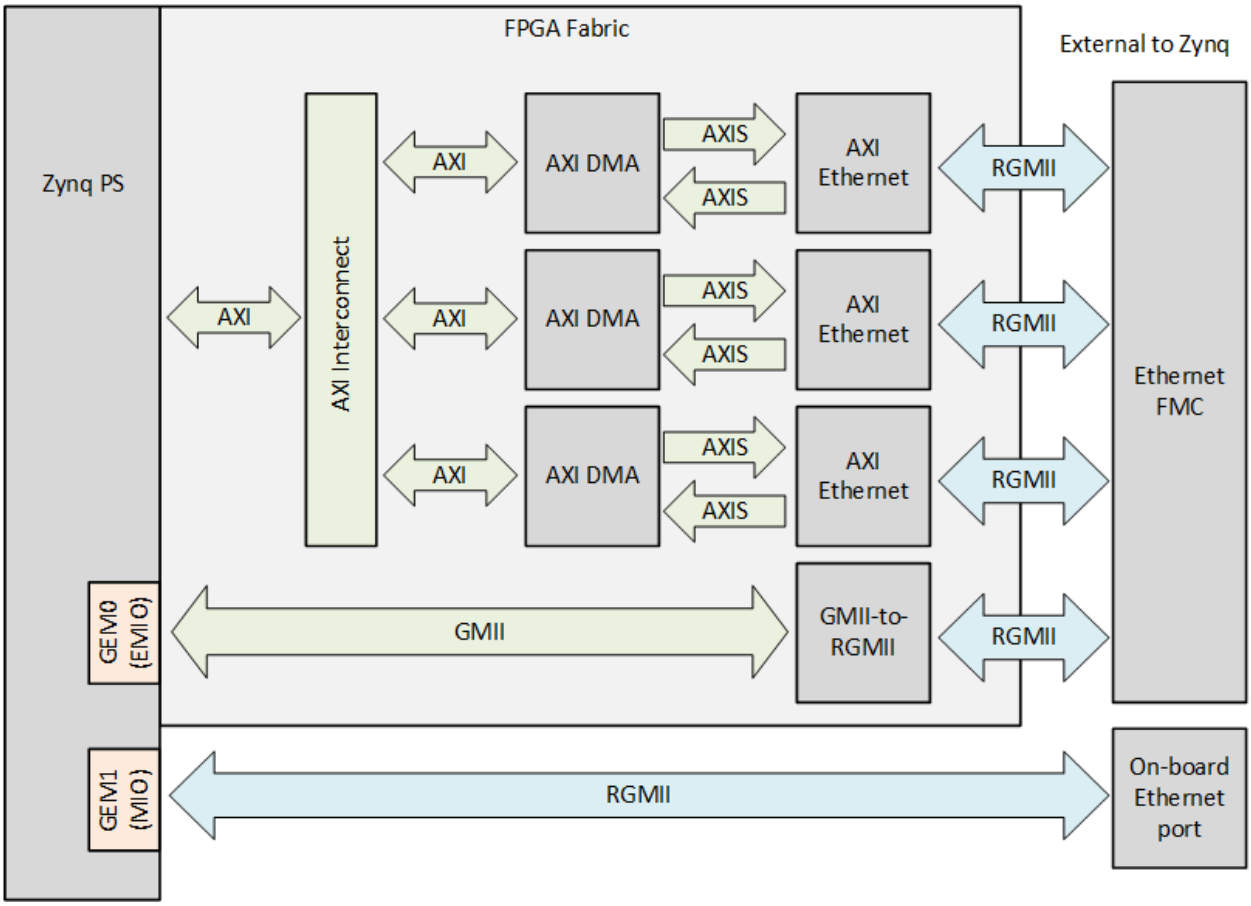

Fig. 1.1: Zynq GEM design block diagram

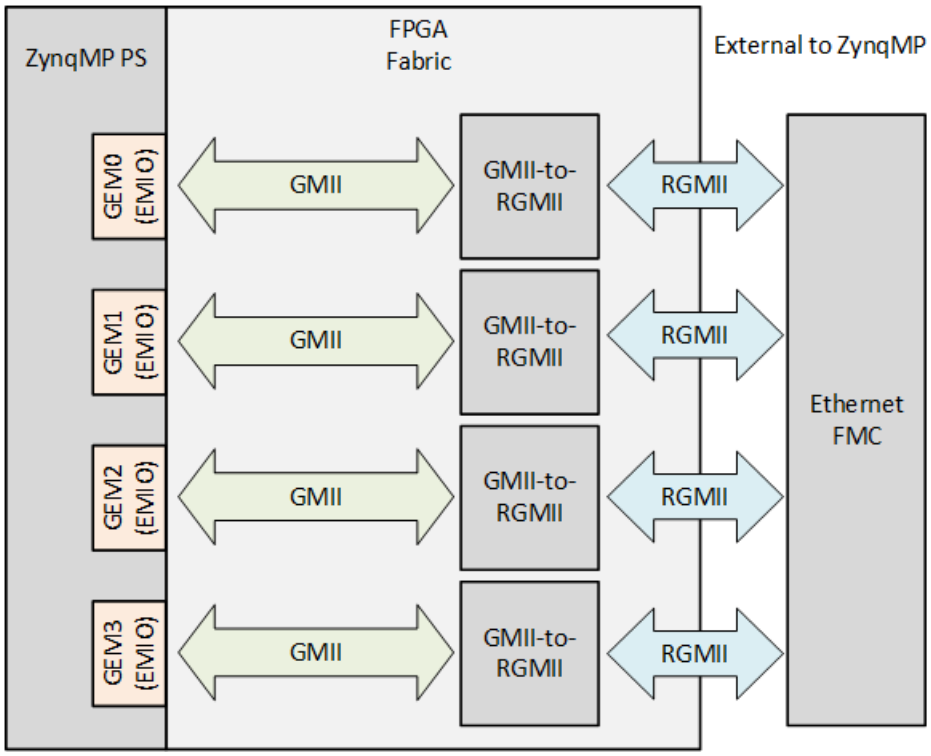

Fig. 1.2: ZynqMP GEM design block diagram

CHAPTER

TWO

REQUIREMENTS

In order to test this design on hardware, you will need the following:

- Vivado 2020.2
- Vitis 2020.2
- PetaLinux Tools 2020.2
- Ethernet FMC
- For the Zynq designs: Xilinx Soft TEMAC license

CHAPTER

THREE

SUPPORTED CARRIER BOARDS

3.1 List of supported boards

Carrier board	FMC connector
Zynq-7000 ZedBoard	LPC
Zynq-7000 MicroZed FMC Carrier with MicroZed 7Z020.	LPC
Zynq-7000 PicoZed FMC Carrier Card V2 with PicoZed 7030	LPC
Zynq-7000 ZC706 Evaluation board	LPC
Zynq UltraScale+ ZCU102 Evaluation board	HPC0 and HPC1 (HPC1 limited to 3 ports)
Zynq UltraScale+ ZCU104 Evaluation board	LPC
Zynq UltraScale+ ZCU106 Evaluation board	HPC0 and HPC1 (HPC1 limited to 2 ports)
Zynq UltraScale+ UltraZed-EG PCIe Carrier Card	LPC
Zynq UltraScale+ UltraZed EV Carrier Card	HPC
Zynq UltraScale+ TE0808-04-09-1EE-S Starter Kit with TE0808 UltraSOM+ MPSoC Module	HPC
Zynq UltraScale+ **`PYNQ-ZU`_** _____	LPC

3.2 Unlisted boards

If you need more information on whether the Ethernet FMC is compatible with a carrier that is not listed above, please first check the compatibility list. If the carrier is not listed there, please contact Opsero, provide us with the pinout of your carrier and we'll be happy to check compatibility and generate a Vivado constraints file for you.

3.3 Board specific notes

3.3.1 ZCU106

- The HPC0 connector on this board supports all 4 ports of the Ethernet FMC. This design uses up all 4 GEMs, leaving the ZCU106's on-board Ethernet port unusable.
- The HPC1 connector only has LA00-LA16 pins connected to the FPGA, therefore our example can only support 2 ports of the Ethernet FMC. The ZCU106's on-board Ethernet port connects to GEM3 and is usable in this design.

3.3.2 ZCU102

- This design supports the ZCU102 Rev 1.0 board. Use a commit before 2017/02/13 for the older Rev-D board design. Note that the FMC pinouts differ between Rev 1.0 and Rev D: https://www.xilinx.com/support/answers/68050.html
- The HPC0 design uses 4x GEMs to connect to ports 0-3 of the Ethernet FMC. This design uses up all 4 GEMs, leaving the ZCU102's on-board Ethernet port unusable.
- The HPC1 design uses 3x GEMs to connect to ports 0-2 of the Ethernet FMC. The 4th port is left unconnected because certain pins required by the Ethernet FMC (namely LA30, LA31 and LA32) are left unconnected on the HPC1 connector of the ZCU102 board. The ZCU102's on-board Ethernet port connects to GEM3 and is usable in this design.

3.3.3 UltraZed-EG and UltraZed-EV

- The UltraZed designs use 4x GEMs to connect to ports 0-3 of the Ethernet FMC. These designs use up all 4 GEMs, leaving the on-board Ethernet port unusable.

3.3.4 Trenz TE0808 Starter Kit

The base board TEBF0808 has a DIP switch that must be set correctly to enable VADJ of 1.8V. Set S5-4 to ON in order to set VADJ to 1.8V.

This design uses up all 4 GEMs, leaving the TEBF0808's on-board Ethernet port unusable.

3.3.5 ZedBoard, MicroZed and PicoZed

When changing `ETH_FMC_PORT` from 0-2 to 3 (ie. when switching to GEM1), it has been noticed that you have to power cycle the board. When the SDK project is configured for AXI Ethernet, it must make some Zynq configurations that are not compatible with the GEM1 configuration.

The on-board Ethernet port on all of these designs is connected to GEM0 and is usable.

3.3.6 Installation of MicroZed, PicoZed, UltraZed, TE0808 board definition files

To use the projects for the MicroZed, PicoZed and UltraZed, you must first install the board definition files for those boards into your Vivado and Xilinx SDK installation.

The following folders contain the board definition files and can be found in this project repository at this location:

https://github.com/fpgadeveloper/ethernet-fmc-zynq-gem/tree/master/Vivado/boards/board_files

- `microzed_7020`
- `picozed_7030_fmc2`
- `ultrazed_3eg_pciecc`
- `TE0808_9EG_1E`

Copy those folders and their contents into the `C:\Xilinx\Vivado\2020.2\data\boards\board_files` folder (this may be different on your machine, depending on your Vivado installation directory). You also need to make a copy into the Xilinx SDK installation at this location: `C:\Xilinx\SDK\2020.2\data\boards\board_files`.

CHAPTER

FOUR

BUILD INSTRUCTIONS

4.1 Source code

The source code for the reference designs is managed on this Github repository: https://github.com/fpgadeveloper/ethernet-fmc-zynq-gem

4.2 Windows users

1. Download the repo as a zip file and extract the files to a directory on your hard drive –OR– Git users: clone the repo to your hard drive
2. Open Windows Explorer, browse to the repo files on your hard drive.
3. In the Vivado directory, you will find multiple batch files (.bat). Double click on the batch file that is appropriate to your hardware, for example, double-click `build-zedboard.bat` if you are using the ZedBoard. This will generate a Vivado project for your hardware platform.
4. Run Vivado and open the project that was just created.
5. Click Generate bitstream.
6. When the bitstream is successfully generated, select *File->Export->Export Hardware*. In the window that opens, tick "Include bitstream" and "Local to project".
7. Return to Windows Explorer and browse to the Vitis directory in the repo.
8. Double click the `build-vitis.bat` batch file. The batch file will run the `build-vitis.tcl` script and build the Vitis workspace containing the hardware design and the software application.
9. Run Xilinx Vitis and select the workspace to be the Vitis directory of the repo.
10. Connect and power up the hardware.
11. Open a Putty terminal to view the UART output.
12. In Vitis, select *Xilinx Tools->Program FPGA*.
13. Right-click on the application and select *Run As->Launch on Hardware (Single Application Debug)*

4.3 Linux users

1. Download the repo as a zip file and extract the files to a directory on your hard drive –OR– Git users: clone the repo to your hard drive
2. Launch the Vivado GUI.
3. Open the Tcl console from the Vivado welcome page. In the console, `cd` to the repo files on your hard drive and into the Vivado subdirectory. For example: `cd /media/projects/ethernet-fmc-zynq-gem/Vivado`.
4. In the Vivado subdirectory, you will find multiple Tcl files. To list them, type `exec ls {*}[glob *.tcl]`. Determine the Tcl script for the example project that you would like to generate (for example: `build-zedboard.tcl`), then `source` the script in the Tcl console: For example: `source build-zedboard.tcl`
5. Vivado will run the script and generate the project. When it's finished, click Generate bitstream.
6. When the bitstream is successfully generated, select *File->Export->Export Hardware*. In the window that opens, tick "Include bitstream" and "Local to project".
7. To build the Vitis workspace, open a Linux command terminal and `cd` to the Vitis directory in the repo.
8. The Vitis directory contains the `build-vitis.tcl` script that will build the Vitis workspace containing the hardware design and the software application. Run the build script by typing the following command: `<path-of-xilinx-vitis>/bin/xsct build-vitis.tcl`. Note that you must replace `<path-of-xilinx-vitis>` with the actual path to your Xilinx Vitis installation.
9. Run Xilinx Vitis and select the workspace to be the Vitis subdirectory of the repo.
10. Connect and power up the hardware.
11. Open a Putty terminal to view the UART output.
12. In Vitis, select *Xilinx Tools->Program FPGA*.
13. Right-click on the application and select *Run As->Launch on Hardware (Single Application Debug)*

CHAPTER

FIVE

STAND-ALONE LWIP ECHO SERVER

These reference designs can be used with the stand-alone lwIP echo server application template that is part of Vitis; however, some modifications are required. The lwIP library needs some modifications to be able to properly configure the Marvell PHYs (88E1510) that are on the Ethernet FMC. The `Vitis` directory of the source repository contains a script that can be used to setup a Vitis workspace containing the echo server application and the modified lwIP library.

The build script does the following:

1. Creates a Vitis workspace in the `Vitis` directory of the source repository.
2. Creates a subdirectory called `embeddedsw` to be used as a local software repository containing the modified lwIP library.
3. Copies the sources from the `EmbeddedSw` directory of the repository to the local software repository (`embeddedsw`), then copies any remaining/unmodified sources from the Vitis installation directory into the local software repository.
4. Generates a lwIP Echo Server example application for each exported Vivado design that is found in the `Vivado` directory. Most users will only have one exported Vivado design.

5.1 Building the Vitis workspace

To build the Vitis workspace and echo server application, you must first generate the Vivado project hardware design (the bitstream) and export the hardware. Once the bitstream is generated and exported, then you can build the Vitis workspace using the provided `Vitis/build-vitis.tcl` script.

5.1.1 Windows users

To build the Vitis workspace, Windows users can run the `build-vitis.bat` file which launches the Tcl script.

5.1.2 Linux users

Linux users must use the following commands to run the build script:

```
cd <path-to-repo>/Vitis
/<path-to-xilinx-tools>/Vitis/2020.2/bin/xsct build-vitis.tcl
```

5.2 Run the application

1. Open Xilinx Vitis.
2. Power up your hardware platform and ensure that the JTAG is connected properly.
3. In the Vitis Explorer panel, double-click on the System project that you want to run - this will reveal the applications contained in the project. The System project will have the postfix "_system".
4. Now click on the application that you want to run. It should have the postfix "_echo_server".
5. Select the option "Run Configurations" from the drop-down menu contained under the Run button on the toolbar (play symbol).
6. Double-click on "Single Application Debug" to create a run configuration for this application. Then click "Run".

The run configuration will first program the FPGA with the bitstream, then load and run the application. You can view the UART output of the application in a console window.

5.3 UART settings

To receive the UART output of this standalone application, you will need to connect the USB-UART of the development board to your PC and run a console program such as Putty. The following UART settings must be used:

- 115200 baud

5.4 IP address

By default, the echo server attempts to obtain an IP address from a DHCP server. This is useful if the echo server is connected to a network. Once the IP address is obtained, it is printed out in the UART console output.

If instead the echo server is connected directly to a PC, the DHCP attempt will fail and the echo server's IP address will default to 192.168.1.10. To be able to communicate with the echo server from the PC, the PC should be configured with a fixed IP address on the same subnet, for example: 192.168.1.20.

5.5 Change the targetted port

The echo server example design currently can only target one Ethernet port at a time. Selection of the Ethernet port can be changed by modifying the defines contained in the `platform_config.h` file in the application sources. Set `PLATFORM_EMAC_BASEADDR` to one of the following values:

5.5.1 ZedBoard, MicroZed and PicoZed designs

- Ethernet FMC Port 0: `XPAR_AXIETHERNET_0_BASEADDR`
- Ethernet FMC Port 1: `XPAR_AXIETHERNET_1_BASEADDR`
- Ethernet FMC Port 2: `XPAR_AXIETHERNET_2_BASEADDR`
- Ethernet FMC Port 3: `XPAR_XEMACPS_1_BASEADDR`
- On-board Ethernet port: `XPAR_XEMACPS_0_BASEADDR`

5.5.2 ZCU102 design (HPC0)

- Ethernet FMC Port 0: `XPAR_XEMACPS_0_BASEADDR`
- Ethernet FMC Port 1: `XPAR_XEMACPS_1_BASEADDR`
- Ethernet FMC Port 2: `XPAR_XEMACPS_2_BASEADDR`
- Ethernet FMC Port 3: `XPAR_XEMACPS_3_BASEADDR`
- ZCU102's on-board Ethernet port: Not usable

5.5.3 ZCU102 design (HPC1)

- Ethernet FMC Port 0: `XPAR_XEMACPS_0_BASEADDR`
- Ethernet FMC Port 1: `XPAR_XEMACPS_1_BASEADDR`
- Ethernet FMC Port 2: `XPAR_XEMACPS_2_BASEADDR`
- ZCU102's on-board Ethernet port: `XPAR_XEMACPS_3_BASEADDR`

5.5.4 ZCU106 design (HPC0)

- Ethernet FMC Port 0: `XPAR_XEMACPS_0_BASEADDR`
- Ethernet FMC Port 1: `XPAR_XEMACPS_1_BASEADDR`
- Ethernet FMC Port 2: `XPAR_XEMACPS_2_BASEADDR`
- Ethernet FMC Port 3: `XPAR_XEMACPS_3_BASEADDR`
- ZCU106's on-board Ethernet port: Not usable

5.5.5 ZCU106 design (HPC1)

- Ethernet FMC Port 0: `XPAR_XEMACPS_0_BASEADDR`
- Ethernet FMC Port 1: `XPAR_XEMACPS_1_BASEADDR`
- Ethernet FMC Port 2: Not usable
- ZCU106's on-board Ethernet port: `XPAR_XEMACPS_3_BASEADDR`

5.5.6 UltraZed, TE0808 design

- Ethernet FMC Port 0: `XPAR_XEMACPS_0_BASEADDR`
- Ethernet FMC Port 1: `XPAR_XEMACPS_1_BASEADDR`
- Ethernet FMC Port 2: `XPAR_XEMACPS_2_BASEADDR`
- Ethernet FMC Port 3: `XPAR_XEMACPS_3_BASEADDR`
- On-board Ethernet port: Not usable

5.5.7 BSP Setting

- When using ports that use AXI Ethernet IP, the BSP setting `use_axieth_on_zynq` must be set to 1.
- When using ports that use Zynq GEM, the BSP setting `use_axieth_on_zynq` must be set to 0.

To change BSP settings: right click on the BSP and click `Board Support Package Settings` from the context menu.

5.6 Example usage

5.6.1 Ping the port

The echo server can be "pinged" from a connected PC, or if connected to a network, from another device on the network. The UART console output will tell you what the IP address of the echo server is. To ping the echo server, use the `ping` command from a command console of a PC that is connected to the echo server (either directly or via network).

Example command: `ping 192.168.1.10`

5.6.2 Connect with telnet

We can also connect to the echo server using telnet and confirm that it is sending back (echoing) the data that we are sending it. From the command prompt of a PC on the same network as the echo server, run the following command:

Example command: `telnet 192.168.1.10 7`

The first argument of the telnet command specifies the IP address of the device to connect to (in our case the echo server). The last argument in the command specifies the port number, which should be 7 for the echo server.

In the blank screen that opens after running the command, you can type letters and they will be sent to the echo server and be echoed back.

5.7 Patches

5.7.1 TEBF0808 ZynqMP FSBL patch

The FSBL for the TEBF0808 board needs some modifications to enable certain clocks before the bitstream is loaded and application/OS is launched. To incorporate the modifications, this repo contains a modified version of the ZynqMP FSBL in the `EmbeddedSw` directory. The Vitis build script sets the `EmbeddedSw` directory as a local software repository, thus all ZynqMP applications in the workspace use the modified ZynqMP FSBL. However, the TEBF0808 specific modifications are braced with `#if defined` statements so that they are only applied to the TEBF0808 design.

5.7.2 UltraZed EG psu_init patch

Since tool version 2020.1, the psu_init sequence contains a function called `serdes_illcalib()` which hangs when executed on the UltraZed EG SoM. We have not yet been able to determine why this function hangs on the UltraZed EG SoM, but the current workaround is to bypass the function call.

To patch the `psu_init.c` file containing the function call, we've added code to the `build-vitis.tcl` script to modify the file after it has been automatically generated by Vitis. If you manually create a platform project for the UltraZed EG project (uz_pci_qgige), you will have to manually open the `psu_init.c` file and comment out the function call to `serdes_illcalib()`.

CHAPTER

SIX

PETALINUX

PetaLinux can be built for these reference designs by using the script in the `PetaLinux` directory of the repository.

6.1 How to build

6.1.1 Requirements

- Windows or Linux PC with Vivado installed
- Linux PC or virtual machine with PetaLinux installed

6.1.2 Instructions

1. First generate the Vivado project hardware design(s) (the bitstream) and export the design(s).
2. Launch PetaLinux by sourcing the `settings.sh` bash script, eg: `source <path-to-installed-petalinux>/settings.sh`
3. Build the PetaLinux project(s) by executing the `build-petalinux` script in Linux.

The script will generate a separate PetaLinux project for all of the generated and exported Vivado projects that it finds in the Vivado directory of this repo.

6.2 UNIX line endings

The scripts and files in the PetaLinux directory of this repository must have UNIX line endings when they are executed or used under Linux. The best way to ensure UNIX line endings, is to clone the repo directly onto your Linux machine. If instead you have copied the repo from a Windows machine, the files will have DOS line endings and you must use the `dos2unix` tool to convert the line endings for UNIX.

1. Copy the cloned repository from your Windows machine to your Linux machine.
2. Use the `cd` command to navigate to the copied repository on your Linux machine.
3. Type `find . -type f -exec dos2unix --keepdate {} +` to convert all of the files to the Unix format.

6.3 How the script works

The PetaLinux directory contains a `build-petalinux` shell script which can be run in Linux to automatically generate a PetaLinux project for each of the generated/exported Vivado projects in the Vivado directory.

When executed, the build script searches the Vivado directory for all projects containing a `.xsa` exported hardware design file. Then for every exported project, the script does the following:

1. Verifies that the `.bit` file exists.
2. Determines the CPU type: Zynq or ZynqMP. It does this by reading the Vivado project file.
3. Creates a PetaLinux project, referencing the exported hardware design (.xsa).
4. Copies the relevant configuration files from the `src` directory into the created PetaLinux project.
5. Builds the PetaLinux project.
6. Generates a BOOT.BIN, boot.scr and image.ub file.

6.4 Launch PetaLinux on hardware

6.4.1 Via JTAG

To launch the PetaLinux project on hardware via JTAG, connect and power up your hardware and then use the following commands in a Linux command terminal:

1. Change current directory to the PetaLinux project directory: `cd <petalinux-project-dir>`
2. Download bitstream to the FPGA: `petalinux-boot --jtag --fpga` Note that you don't have to specify the bitstream because this command will use the one that it finds in the `./images/linux` directory.
3. Download the PetaLinux kernel to the FPGA: `petalinux-boot --jtag --kernel`

6.4.2 Via SD card

To launch the PetaLinux project on hardware via SD card, copy the following files to the root of the SD card:

- `/<petalinux-project>/images/linux/BOOT.bin`
- `/<petalinux-project>/images/linux/boot.scr`
- `/<petalinux-project>/images/linux/image.ub`

Then connect and power your hardware.

6.5 Configuration files

The configuration files contained in the *src* directory include:

- Device tree
- Rootfs configuration (to include ethtool)
- Interface initializations (sets eth0-4 interfaces to DHCP)
- Kernel configuration

- Board BSPs
- Patches (see below)

6.5.1 ZCU104 ZynqMP FSBL patch

The FSBL for Zynq Ultrascale+ needs a patch to properly enable VADJ on the ZCU104 board. The FSBL has code to read the FMC card's EEPROM and then enable VADJ to the correct value. The code in fact reads from the ZCU104 board's EEPROM and not the FMC's EEPROM. It also only reads 32 bytes from the EEPROM, which is not sufficient to include the VADJ voltage data. For both of these reasons, the FSBL does not properly enable VADJ on this board.

There is a patch contained in this repo that fixes this issue so that VADJ is correctly enabled on the ZCU104 board.

6.5.2 TEBF0808 ZynqMP FSBL patch

The FSBL for the TEBF0808 board needs some modifications to enable certain clocks before the bitstream is loaded and application/OS is launched. To incorporate the modifications into the PetaLinux build, this repo contains a patch for the FSBL located here:

`<repo>/PetaLinux/src/tebf0808/project-spec/meta-user/recipes-bsp/fsbl/files/tebf0808_fsbl.patch`

6.6 Port configurations

6.6.1 MicroZed, PicoZed, ZC702, ZC706, ZedBoard

- eth0: Ethernet port of the dev board (GEM0)
- eth1: Ethernet FMC Port 3 (GEM1)
- eth2: Ethernet FMC Port 0 (AXI Ethernet)
- eth3: Ethernet FMC Port 1 (AXI Ethernet)
- eth4: Ethernet FMC Port 2 (AXI Ethernet)

6.6.2 ZCU104, ZCU102 (HPC0), ZCU106 (HPC0) and TEBF0808

- eth0: Ethernet FMC Port 0 (GEM0)
- eth1: Ethernet FMC Port 1 (GEM1)
- eth2: Ethernet FMC Port 2 (GEM2)
- eth3: Ethernet FMC Port 3 (GEM3)

Note that the Ethernet port of the dev board in these designs is not connected to any GEM and is thus unusable.

6.6.3 ZCU102 (HPC1)

- eth0: Ethernet FMC Port 0 (GEM0)
- eth1: Ethernet FMC Port 1 (GEM1)
- eth2: Ethernet FMC Port 2 (GEM2)
- eth3: ZCU102 on-board Ethernet port (GEM3)

6.6.4 ZCU106 (HPC1)

- eth0: Ethernet FMC Port 0 (GEM0)
- eth1: Ethernet FMC Port 1 (GEM1)
- eth2: ZCU102 on-board Ethernet port (GEM3)

6.7 Example Usage

6.7.1 Enable port

This example will bring up a port.

```
root@zynqgem:~# ifconfig eth1 up
[  378.871550] pps pps1: new PPS source ptp1
[  378.875583] macb ff0c0000.ethernet: gem-ptp-timer ptp clock registered.
[  382.943505] macb ff0c0000.ethernet eth1: unable to generate target frequency:␣
↪125000000 Hz
[  382.951774] macb ff0c0000.ethernet eth1: link up (1000/Full)
[  382.957441] IPv6: ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
```

6.7.2 Enable port with fixed IP address

This example sets a fixed IP address to a port.

```
root@zynqgem:~# ifconfig eth1 192.168.2.31 up
[  424.839768] pps pps1: new PPS source ptp1
[  424.843798] macb ff0c0000.ethernet: gem-ptp-timer ptp clock registered.
[  428.927505] macb ff0c0000.ethernet eth1: unable to generate target frequency:␣
↪125000000 Hz
[  428.935778] macb ff0c0000.ethernet eth1: link up (1000/Full)
[  428.941450] IPv6: ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
```

6.7.3 Enable port using DHCP

This example enables a port and obtains an IP address for the port via DHCP. Note that the port must be connected to a DHCP enabled router.

```
root@zynqgem:~# udhcpc -i eth1
udhcpc: started, v1.31.0
[  314.831199] macb ff0c0000.ethernet eth1: unable to generate target frequency:␣
↪125000000 Hz
[  314.839489] macb ff0c0000.ethernet eth1: link up (1000/Full)
[  314.845181] pps pps1: new PPS source ptp1
[  314.849205] macb ff0c0000.ethernet: gem-ptp-timer ptp clock registered.
[  314.855955] IPv6: ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
udhcpc: sending discover
udhcpc: sending select for 192.168.2.24
udhcpc: lease of 192.168.2.24 obtained, lease time 259200
RTNETLINK answers: File exists
/etc/udhcpc.d/50default: Adding DNS 192.168.2.1
```

6.7.4 Check port status

In this example, we use the `ifconfig` command with no arguments to check the port status. The first interface (eth1) is connected to the Ethernet FMC port 0 and it has been enabled and configured with IP address 192.168.2.23.

```
root@zynqgem:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:01:22
          inet addr:192.168.2.23  Bcast:192.168.2.255  Mask:255.255.255.0
          inet6 addr: fe80::20a:35ff:fe00:122/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0 frame:0
          TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:794 (794.0 B)  TX bytes:2000 (1.9 KiB)
          Interrupt:30

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

We can also use `ethtool` to check the port status as follows.

```
root@zynqgem:~# ethtool eth0
Settings for eth0:
        Supported ports: [ TP MII FIBRE ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
```

(continues on next page)

(continued from previous page)

```
        Supported pause frame use: Symmetric Receive-only
        Supports auto-negotiation: Yes
        Supported FEC modes: Not reported
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Half 1000baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Advertised FEC modes: Not reported
        Link partner advertised link modes:  10baseT/Half 10baseT/Full
                                             100baseT/Half 100baseT/Full
                                             1000baseT/Full
        Link partner advertised pause frame use: No
        Link partner advertised auto-negotiation: Yes
        Link partner advertised FEC modes: Not reported
        Speed: 1000Mb/s
        Duplex: Full
        Port: MII
        PHYAD: 0
        Transceiver: internal
        Auto-negotiation: on
        Link detected: yes
```

6.7.5 Ping link partner using specific port

In this example we ping the link partner at IP address 192.168.2.10 from interface eth1.

```
root@zynqgem:~# ping -I eth0 192.168.2.10
PING 192.168.2.10 (192.168.2.10): 56 data bytes
64 bytes from 192.168.2.10: seq=0 ttl=128 time=0.448 ms
64 bytes from 192.168.2.10: seq=1 ttl=128 time=0.416 ms
64 bytes from 192.168.2.10: seq=2 ttl=128 time=0.418 ms
64 bytes from 192.168.2.10: seq=3 ttl=128 time=0.409 ms
```

6.8 Known Issues

6.8.1 AXI Ethernet issue on Zynq designs 2020.2

There is an issue in the PetaLinux 2020.2 release that affects the **AXI Ethernet** connected ports on **Zynq** based designs. On these ports, it seems to be necessary to use the following procedure to bring up a port. Note that the interface and IP address were chosen as examples, but this procedure applies to all AXI Ethernet connected ports (eth2, eth3 and eth4) on the Zynq based designs (MicroZed, PicoZed, ZedBoard, ZC702 and ZC706).

```
ifconfig eth0 up
ifconfig eth0 down
ifconfig eth0 192.168.1.10 up
```

In earlier releases, it was only necessary to run the last command to bring up a port. This issue does not affect the Zynq Ultrascale+ based designs. This issue does not affect the stand-alone echo server operation. We have not yet

determined the cause of this issue but if you have any information, please let us know.

CHAPTER

SEVEN

UPDATING THE PROJECTS

This section contains instructions for updating the reference designs. It is intended as a guide for anyone wanting to attempt updating the designs for a tools release that we do not yet support. Note that the update process is not always straight-forward and sometimes requires dealing with new issues or significant changes to the functionality of the tools and/or specific IP. Unfortunately, we cannot always provide support if you have trouble updating the designs.

7.1 Vivado projects

1. Download and install the Vivado release that you intend to use.
2. If you are using one of the following boards, you will have to download and install the latest board files for that target platform. Other boards are already built into Vivado and require no extra installation.
   - MicroZed board files can be downloaded here
   - PicoZed board files can be downloaded here
   - UltraZed EG and EV board files can be downloaded here
3. In a text editor, open the `Vivado/build-<target>.bat` file for the design that you wish to update, and perform the following changes:
   - Update the tools version number to the one you are using (eg. 2020.2)
4. In a text editor, open the `Vivado/build-<target>.tcl` file for the design that you wish to update, and perform the following changes:
   - Update the `version_required` variable value to the tools version number that you are using.
   - Update the year in all references to `Vivado Synthesis <year>` to the tools version number that you are using. For example, if you are using tools version 2020.2, then the `<year>` should be 2020.
   - Update the year in all references to `Vivado Implementation <year>` to the tools version number that you are using. For example, if you are using tools version 2020.2, then the `<year>` should be 2020.
   - If the version of the board files for your target platform has changed, update the `board_part` parameter value to the new version.

After following the above steps, you can now run the build script. If there were no significant changes to the tools and/or IP, the build script should succeed and you will be able to open and generate a bitstream for the Vivado project.

7.2 PetaLinux

The main procedure for updating the PetaLinux project is to update the BSP for the target platform. The BSP files for each supported target platform are contained in the `PetaLinux/src` directory. For example, the BSP files for the ZedBoard are located in `PetaLinux/src/zedboard`.

1. Download and install the PetaLinux release that you intend to use.
2. Download and install the BSP for the target platform for the release that you intend to use.
   - For ZC706, ZCU102, ZCU104, ZCU106 and ZedBoard, download the BSP from the PetaLinux download page
   - For MicroZed and PicoZed, download the BSP for the **ZedBoard** from the PetaLinux download page
   - For TEBF0808, UltraZed EG and EV, download the BSP for the **ZCU102** from the PetaLinux download page
3. Update the BSP files for the target platform in the `PetaLinux/src/<platform>` directory. These are the specific directories to update:
   - `<platform>/project-spec/configs/*`
   - `<platform>/project-spec/meta-user/*`

   The simple way to update the files is to delete those in the repository and copy in those from the BSP that you just downloaded.
4. Apply the required modifications to the updated BSP files. The modifications are described for each target platform in the following sections.

7.2.1 Change project name

This BSP modification applies to all target platforms.

1. Append the following lines to `project-spec/configs/config`:

```
# Set project name
CONFIG_SUBSYSTEM_HOSTNAME="zynqgem"
CONFIG_SUBSYSTEM_PRODUCT="zynqgem"
```

Note that this will set the project name to "zynqgem" but you can use a more descriptive name, for example one that includes the target platform name and the tools version.

7.2.2 Add tools to root filesystem

This BSP modification applies to all target platforms.

1. Append the following lines to `project-spec/configs/rootfs_config`:

```
# Useful tools for Ethernet FMC
CONFIG_ethtool=y
CONFIG_ethtool-dev=y
CONFIG_ethtool-dbg=y
CONFIG_iperf3=y
```

2. Append the following lines to `project-spec/meta-user/conf/user-rootfsconfig`:

```
CONFIG_iperf3
CONFIG_ethtool
```

7.2.3 Include port config in device tree

This BSP modification applies to all target platforms.

1. Append the following line after `/include/ "system-conf.dtsi"` in `project-spec/meta-user/recipes-bsp/device-tree/files/system-user.dtsi`:

```
/include/ "port-config.dtsi"
```

2. Append the following line after `SRC_URI += "file://system-user.dtsi"` in `project-spec/meta-user/recipes-bsp/device-tree/device-tree.bbappend`:

```
SRC_URI += "file://port-config.dtsi"
```

7.2.4 Add kernel configs

This BSP modification applies to all target platforms.

1. Add the following lines to the top of file `project-spec/meta-user/recipes-kernel/linux/linux-xlnx/bsp.cfg`:

```
# Required by all designs
CONFIG_XILINX_GMII2RGMII=y

# Required by BSP
```

7.2.5 Mods for ZCU104

These modifications are specific to the ZCU104 BSP.

1. Append the following lines to `project-spec/configs/config`:

```
# ZCU104 port configs

CONFIG_SUBSYSTEM_ETHERNET_PSU_ETHERNET_0_SELECT=y
CONFIG_SUBSYSTEM_ETHERNET_PSU_ETHERNET_1_SELECT=n
CONFIG_SUBSYSTEM_ETHERNET_PSU_ETHERNET_2_SELECT=n
CONFIG_SUBSYSTEM_ETHERNET_PSU_ETHERNET_3_SELECT=n
CONFIG_SUBSYSTEM_ETHERNET_PSU_ETHERNET_0_USE_DHCP=y
CONFIG_SUBSYSTEM_ETHERNET_PSU_ETHERNET_1_USE_DHCP=y
CONFIG_SUBSYSTEM_ETHERNET_PSU_ETHERNET_2_USE_DHCP=y
CONFIG_SUBSYSTEM_ETHERNET_PSU_ETHERNET_3_USE_DHCP=y
CONFIG_SUBSYSTEM_ETHERNET_PSU_ETHERNET_0_MAC="00:0a:35:00:22:01"
CONFIG_SUBSYSTEM_ETHERNET_PSU_ETHERNET_1_MAC="00:0a:35:00:22:02"
CONFIG_SUBSYSTEM_ETHERNET_PSU_ETHERNET_2_MAC="00:0a:35:00:22:03"
CONFIG_SUBSYSTEM_ETHERNET_PSU_ETHERNET_3_MAC="00:0a:35:00:22:04"
```

2. Add patch for FSBL to `project-spec/meta-user/recipes-bsp/fsbl/`. You will have to update this patch for the version of PetaLinux that you are using. Refer to the existing patch files in that location for guidance.

7.2.6 Mods for ZCU106

These modifications are specific to the ZCU106 BSP.

1. Append the following lines to `project-spec/configs/config`. The first option prevents the removal of the PL DTB nodes that we need in this design. The second option disables the FPGA manager.

```
# ZCU106 configs

CONFIG_SUBSYSTEM_REMOVE_PL_DTB=n
CONFIG_SUBSYSTEM_FPGA_MANAGER=n
```

CHAPTER

EIGHT

TROUBLESHOOTING

8.1 Build failures

Check the following if the project fails to build or generate a bitstream:

1. **Are you using the correct version of Vivado for this version of the repository?**
   Check the version specified in the Requirements section of the README.md file. Note that this project is regularly updated and you may have to refer to an earlier commit of this repo if you are using an older version of Vivado.
2. **Did you correctly follow the build instructions?**
   Please check the build instructions carefully as you may have missed a step.
3. **Did you copy/clone the repo into a short directory structure?**
   Windows doesn't cope well with long directory structures, so copy/clone the repo into a short directory structure such as `C:\projects\`. When working in long directory structures, you can get errors relating to missing files, particularly files that are normally generated by Vivado (FIFOs, etc).

8.2 PetaLinux issues

8.2.1 Ports not working

Check the following if you are unable to get ports working in PetaLinux.

1. **Check the interface-to-port assignment for your design**
   The assignment of interfaces (eg. eth0, eth1, eth2, etc) to ports (eg. Ethernet FMC port 0, 1, 2 and 3) is specific to the design that you are using. The interface to port assignment is documented here.
2. **Each port must be assigned to a different subnet**
   If you assign interface eth0 to IP address 192.168.1.10, then you must use a different subnet for the IP address of eth1, eth2 and eth3. Multiple ports that are managed under Linux must be assigned to different subnets, or they will not work. An example address assignment would be eth0=192.168.1.10, eth1=192.168.2.10, eth2=192.168.3.10, eth3=192.168.4.10.

8.2.2 Dropped pings/packets

No dropped packets are to be expected with our example designs. If you are experiencing dropped packets of any kind, this can be an indication of one of the following issues:

- Timing in the FPGA design is not optimal (check for timing errors in the Vivado design)
- Timing of the RGMII interface is not optimal (can be due to the FPGA design or the PHY configuration)

We ensure that there are no timing errors or issues on all of our designs before making a release, so typically this problem occurs on custom designs where the timing issues have not yet been optimized. Please contact us for support if you are experiencing dropped packets.

CHAPTER

NINE

REVISION HISTORY

This is the first version of the documentation.