X2O Chip2Chip example project notes

A. Madorsky, University of Florida/Physics

DRAFT 2021-02-03

# Introduction

The ZYNQ device on the X2O power module is using AXI Chip2Chip links to provide fast communication path to the payload modules. Each link can use up to two GMT lanes, with the top bit rate limited by ZYNQ device at 3.75 Gbps.

# Example project

An example project is provided as a reference or a starting point for user’s projects.

Table 1 shows the example project details.

|  |  |
| --- | --- |
| **FPGA family** | Ultrascale+ |
| **Partnumber** | xcku15p-ffva1760-2-e |
| **MGT type** | GTH |
| **Git link** | <https://github.com/madorskya/apex_ku15p.git> |
| **Project folder** | apex\_ku15p\_c2c\_mgt |
| **Git tag** | v1.2 |

Table . Example project details

Example project design structure is shown in Figure 1.

AXI

Inter-connect

BRAM 0

Chip2Chip IP

Figure . Example project structure

MGT wrapper

To ZYNQ

MGT

C2c\_adapter

BRAM 1

Table 2 below lists the components of the example design and shows some details about them.

|  |  |
| --- | --- |
| **Component** | **Notes** |
| MGT | Generated using standard Xilinx Transceiver Wizard. See important details of the settings in Table 3. |
| MGT wrapper | Created from the example design provided by Transceiver Wizard, with minimal modifications. Name of this module in the example design is: c2c\_gth\_tux. |
| C2c\_adapter | Custom module connecting Chip2Chip AXI stream RX and TX interfaces to the MGT ports. |
| Chip2Chip IP | Standard Xilinx Chip2Chip IP generated in Aurora 8b/10b configuration. However, note that this example design is not using Aurora IPs. |
| AXI interconnect | Standard Xilinx AXI interconnect IP. This interconnect can be expanded as needed for user’s logic. |
| BRAM 0 and 1 | Standard Xilinx BRAM controllers and BRAM IPs. These modules are provided for memory access tests, and can be replaced with user’s logic as needed. |

Table . Example design components.

The example design is using GTH MGT from Ultrascale+ family. If you need to generate your own MGT IP for another MGT type, please pay attention to the settings listed in Table 3.

|  |  |
| --- | --- |
| **MGT parameter** | **Details** |
| Lane count | 1 |
| Bit rate | 3.75 Gbps |
| Encoding | 8b/10b |
| PLL type | CPLL |
| User data width | 32 bits |
| Comma alignment symbol | K28.5, a.k.a. IDLE, plus only, bit sequence = 0101111100,  mask = 1111111111 |
| Alignment boundary | Four-byte boundary |
| Equalization mode | LPM preferable (if available) |
| Clock correction | Single sequence, length = 4 bytes. Patterns:   |  |  | | --- | --- | | Pattern 0 | 10111100 | | Pattern 1 | 01010000 | | Pattern 2 | 00000000 | | Pattern 3 | 00000000 |   Keep Idle = Disable  Minimum repetition = 0 |
| Include in the example design | Simple transmitter user clocking network  Simple receiver user clocking network  Reset controller |

Table 3. MGT settings details

# Procedure for MGT wrapper generation

The procedure for MGT wrapper generation is outlined below. Please note that this is only necessary if the MGT type you’re using is not GTH from Ultrascale+ family. If it is, you can use the MGT wrapper provided in the X2O example design without changes.

1. Generate MGT as shown in Table 3.
2. Generate example design for that MGT.
3. Include the entire example design into your project. The top-level file from the example design becomes the MGT wrapper.
4. Rework MGT wrapper as shown below:
   1. Remove pattern generators and checkers that are included by Transceiver Wizard.
   2. Make sure the example design is using TXOUTCLK to drive fabric TX and RX clocks
   3. Instantiate c2c\_adapter module inside MGT wrapper, wire it to the MGT TX and RX as shown in the X2O example design.
   4. Connect the MGT wrapper IO to the block design following X2O example design.

# Tests with X2O power board

The details of the ZYNQ firmware and software are listed in Table 4.

|  |  |
| --- | --- |
| **FPGA family** | 7 |
| **Partnumber** | xc7z015clg485-1 |
| **MGT type** | GTP |
| **Git link** | <https://github.com/madorskya/apex.git> |
| **Project folder** | control/apex\_control\_mgt |
| **Bit stream file** | control/apex\_control\_mgt/apex\_control\_mgt.runs/impl\_1/apex\_control\_mgt\_top.bit |
| **Software folder** | control/soft |
| **Git tag** | v1.2 |

Table 4. ZYNQ firmware and software details

## Test procedure:

1. Connect your payload FPGA board to the bottom Firefly connector on the X2O power board.
2. **ssh** into the power board
3. Remove everything in **/root** directory
4. Rework **eth1\_link\_up.sh** file in the git software folder to set your preferred IP address
5. Copy the entire contents of the Software folder (Table 4) into **/root**
6. Mount the SD card BOOT partition: **./mnt.sh**
7. Copy the bit file (Table 4) into **/mnt/boot/7z015.bit**
8. **reboot**
9. Turn on the payload power: **./payload\_on.sh.** If your payload board has independent power, turn it on at this time.
10. Configure payload board clocks. This step depends on your particular board. The example clock configuration script for UF KU15P board is here: **./payload\_clocks.sh**
11. If necessary, start the XVC server so you can access your payload FPGA: **./xvc\_jtag.sh**
12. Load firmware into payload FPGA. You can do it via XVC, or by using JTAG cable.
13. Reset the AXI link: **./c2c\_reset.sh**. Check that it reports **link\_stat\_bot = 0x1**. This means the AXI link is operational
14. **cd c2c**
15. Test the link by running **c2c\_memcpy bot\_bram0**. This program is writing and reading BRAM0 on in the payload FPGA in an infinite loop. It will report comparison errors and bus errors (if any).

# Revisions

|  |  |
| --- | --- |
| **Date** | **Changes** |
| 2021-03-03 | Initial creation |