Aurora 8b/10b on Polarfire
Reference Design User Guide

A.L.S.E
8, Passage Barrault
75013 PARIS
France
Tel : +33 1 84 16 32 32

1. REVISION HISTORY

The following table shows the revision history for this document and the associated IP.

Version	Date	Revision
1.0	Nov 2023	Initial Version (for Polarfire)
1.1	Feb 2024	Add compilation instructions with Aurora netlist
1.2	April 2024	Fixed minor type, details about CRC activation.

Table 1: Revision history

2. GENERAL INFORMATION

This document describes the content of ALSE Aurora 8B/10B loopback Reference Design for Polarfire.
This design is built for the Polarfire FPGA Evalutation Kit.

Loopback is done over 1 lane, using the only SFP+ connector of the board. It is configured to work at 3.125 Gbits/s.

This Reference Design does re-transmit any frame received on its RX side to its TX side.
PDU (Data) frames as well as NFC and UFC Flow Control frames are supported.

With this design, we demonstrate the full compliance of our implementation with the Xilinx Aurora 8b/10b protocol, allowing interoperability with Xilinx FPGAs through Xilinx Aurora 8b/10b IP, as well as with Intel or Lattice boards using our own Aurora IP !

Here is the exact configuration of the IP in the Reference Design described below. Static parameters can be quickly modified to be adapted to your need !

→ Please let ALSE know if you would like to test our design in a different configuration on this board (CRC on, different speed, streaming mode…).

2.1. IP configuration

In the **current** version, the IP is configured as follow :

- ✓ Full-Duplex (of course compulsory for loopback design)
- ✓ 1 x transceiver lane at 3.125 Gbits/s through SFP+ connector.
- ✓ 32 bits (4 Bytes width) user Datapath.
- ✓ Framing interface.
  Note : we have de-activated the CRC to demonstrate the highest bandwidth achieved.
  For some applications, the user can enable the CRC feature to verify automatically the integrity of the data received. If framing mode, activating the CRC while using short frames has a significant impact on the bandwidth.
- ✓ User Flow Control in direct (Xilinx-IP style) mode.
- ✓ Native Flow Control in completion mode.
- ✓ Clock compensation sequence generation enabled.
- ✓ 125 MHz reference clock for transceivers.

2.2. Possible configurations

Here is the list of the **possible configurations** of our IP on **this development kit**:

- ✓ Full-Duplex and Simplex Tx/Rx Operations.
- ✓ Up to 6.6 Gbps (Gigabits per seconds) per lane.
- ✓ SFP+ or SMA connectors. Multiple lanes possible (max 2).
- ✓ 16 bits (2 Bytes width) and 32 bits (4 Bytes width) user datapath.
- ✓ Framing and Streaming interface.
- ✓ User Flow Control.
- ✓ Native Flow Control in immediate and completion mode.
- ✓ Additional CRC for PDU Frames (16bits or 32bits, depending chosen user Datapath)
- ✓ Per lane polarity inversion and skew compensation.

2.1. Delivery content

Here is the list of files available in the Reference Design delivery.

aurora_8b10b	Reference folder
doc	IP related documentation
src*	VHDL source code and IP files (Xilinx and Microchip files)
boards	Reference design files for Intel and Xilinx boards
xilinx/Vivado_VC707	Xilinx VC707 Master Reference Design files
microchip/MPF300-EVAL-KIT	Polarfire loopback design top-level file
common	Common source code
misc	Common source code as well
test_modules	Validation modules (generator / checker for PDU, UFC and NFC streams)
mc_aurora*	Aurora controller sources, including transceivers instantiation.
MPF300*	Microchip IPs TCL import files (TX PLL, transceivers, REF CLK BUFFERS…) used in the transceiver wrapper
xcvr_wrapper.vhd	Transceiver wrapper VHDL entity
mc_aurora_wrapper_*.vhd	Netlist or RTL Aurora controller depending on delivery type.
mc_aurora_top.vhd	Top wrapper, instantiating wrapper and transceiver.
fit	Synthesis files.
microsemi	Project files.
MPF300_EVAL_KIT*.ppd	Binary (.ppd) file for Polarfire Evaluation Development Kit. It is only present if it is not an IP delivery.
.sh, .tcl, .sdc, .pdc	Everything you need to create the full Reference Design, using our IP, including timing constraints and pin assignments.
xilinx	Projects for Xilinx boards

Figure 1: IP package contents

***Note : Folders and files that appear in bold are present only in a IP delivery.**
In an Example Design delivery, only the Xilinx side source code is available, the Microchip design is already compiled in a .ppd and ready to be programmed.

3. ARCHITECTURE

This chapter describes the architecture of the Reference Design provided with the Aurora 8b/10b IP Core.

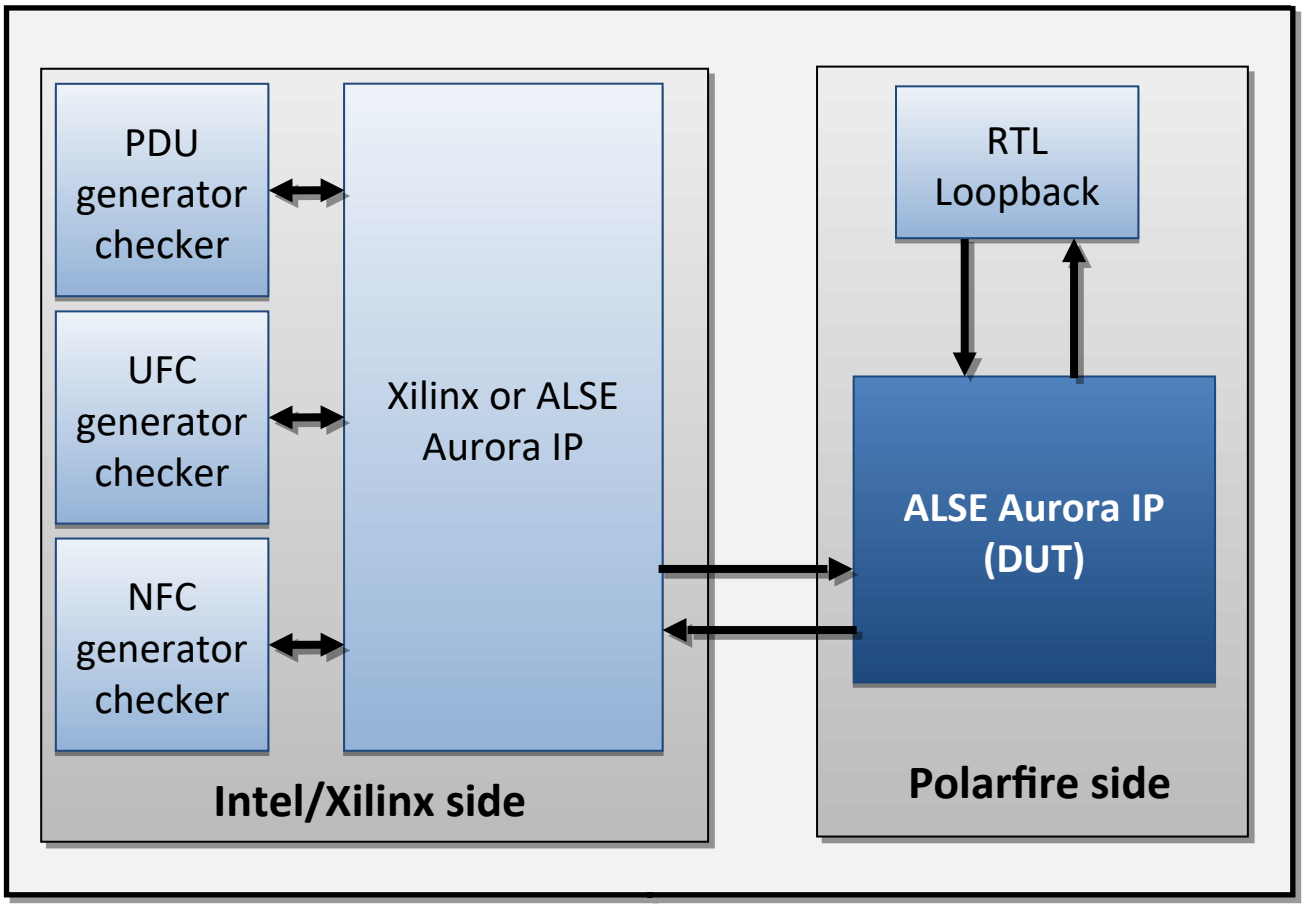


Figure 2: Reference Design architecture

Generator / Checker :

The Generator & Checker modules use LFSRs to generate pseudo random sequences for PDU, UFC and NFC control signals. Since the sizes of the LFSRs are not the same for all signals, most of the use cases are tested by this method.

The Data bytes are incremented ("counter") as they are valid (using the byte enables), so the checker can easily verify the absence of duplicated or lost bytes.

Checking, Generation of results and various Control signals are managed through JTAG using an Intel tool named "In-System Sources & Probes" (see later).

RTL Loopback :

The loopback function in the Polarfire Reference Design does re-send PDU and UFC Data as they are received.

For PDU, a FIFO stores the incoming data from the RX interface and writes these data to the TX interface.

UFC Data are checked using the UFC checker module. As long as UFC frames are correct, new UFC requests are generated on the TX interface. If an error is detected, no request is sent so that the checker at the Intel side will detect the error.

NFC requests are generated in order to avoid overflow on the PDU datapath by looking at the FIFO level.
NFC requests are also re-sent as they are received.

3.1. Microchip Design compilation

This part describes the steps to re-create the Libero Reference Design project using the scripts located in the *fit/* folder.
This is only available for IP delivery (and not with Example Design deliveries).
Libero 2023.2 or above should be installed and accessible in you $PATH variable.
Depending on your OS :
- Linux, execute the steps below.
- Windows, go to 3.1.3.

3.1.1. Under Linux

Open a terminal in the **fit/microsemi/polarfire/** directory.

Run the following command :

```
$ ./gen_project.sh MPF300_EVAL_KIT_loopback
```

This command calls Libero in batch mode through the three TCL scripts located in this folder :

- **gen_project.tcl** : download the required Libero IPs, create the Libero project and add the Reference Design sources. It includes the timing constraint (.sdc) and pin assignments (.pdc) files as well.
  It imports and creates all the Microchip IPs required in the design (including transceivers IPs).
- **import_aurora_xcvr.tcl** : Links the transceiver wrapper file to the project.
- **build_prj.tcl** : Builds the project hierarchy, applies constraints files previously added, and sets the project top-level.

The script has several options but the one presented above in **the only one working with a netlist format delivery**.

3.1.2. Under Windows

Open a terminal (not a power shell) in the **fit\microsemi\polarfire** directory.
make sure **libero.exe is in your path** and that you can invoke it directly in the terminal.

Run the following command :

```
$ gen_project.bat MPF300_EVAL_KIT_loopback
```

This command invokes Libero and runs the same three TCL scripts as described above (for Linux).
The script has several options but the one presented above in **the only one working with a netlist format delivery**.

3.1.3. Project compilation

Steps above created the **MPF300_EVAL_KIT_loopback.prjx project** file that you can now open in Libero.

In the *Design Flow* window, run *Generate Bitstream* to run the full Libero flow.

By clicking on *Manage Constraints*, you can check that our constraints are correct and applied.

In the report window you can check that the compilation is successful and that the resource usage corresponds to what described in the Aurora IP User Guide (depending on your configuration).

You can now program you Polarfire board with the loopback design and test its interoperability with a Xilinx or an Intel master design !

It is possible to change the link speed of the design by modifying the configuration of the **PF_XCVR_ERM** and **PF_TX_PLL** Microchip IPs used in the design. In the *Components* window click on the corresponding IPs to see current configuration.
See Microchip transceiver documentations to get more information on how to configure the transceivers !

3.2. JTAG debugging

The main difference with the simulation is that the hardware implemented version uses JTAG accessible control modules in order to control and monitor the tester/checker.

These modules are accessible through :

- "**In-System Sources and Probes**", a tool available from Intel Quartus Prime.
  Open the file "**insys_src_prb.spf**" in Quartus to control the design.
- "**Virtual Input/Output**" a tool from Xilinx/AMD Vivado.
  Open the file "**debug_nets.ltx**" in Vivado to control the design.

JTAG allows real-time modification of :

- Enable / Disable PDU generation.
- Enable / Disable NFC generation.
- Enable / Disable UFC generation.
- NFC immediate or completion mode.
- PDU pause (some cycles with valid='0') or contiguous (all cycles valid).
- PDU endless (no End Of Packet generated).
- Counters and link Reset. **- INTEL ONLY**

JTAG can also monitor these values :

- Number of PDU frames sent / received.
- Number of UFC frames sent / received.
- Number of NFC requests (pause cycles) sent / received.
- Number of Errors in PDU frames received.
- Number of Errors in UFC frames received.
- PDU bit rate on TX interface (use byte enable to count valid bytes). **- INTEL ONLY**
- PDU bit rate on RX interface (use byte enable to count valid bytes). - INTEL ONLY
- Transceiver reference clock frequencies for Intel FPGA. - INTEL **ONLY**

3.3. LEDS and buttons

The Polarfire loopback design uses the few control LEDs available to output basic debug information.
Debugging and verification is performed from the Master board using the JTAG debug tools presented above.

- LED4 (F22) : **Link-up** information : led is **OFF** when link is **up!**
- LED5 (B26) : **Heartbeat** signal, should blink if design is alive.
- LED6 (C26) : User push-button SWITCH10 (B19) is directly wired to this led.
- LED7 (D25) : POR signal, should be ON when reset is inactive.
- LED8 (C27) : Is transceiver **RX recovery clock alive**? **OFF** when true.
- LED9 to 11 : wired to '1' to check LED polarity. Always ON.

Master boards usually also have some LEDs displaying link up and reset information despite of JTAG tools availability for fast debug. To be checked in the Reference Design top level !

3.4. Master Hardware Tester User Guide

Here we help you to setup the experiment using a Polarfire dev kit and an Intel/Xilinx master board.
Follow the next step to generate an **Intel** master design or 3.3.2 to generate a **Xilinx** master design.

3.4.1. Intel Master Design

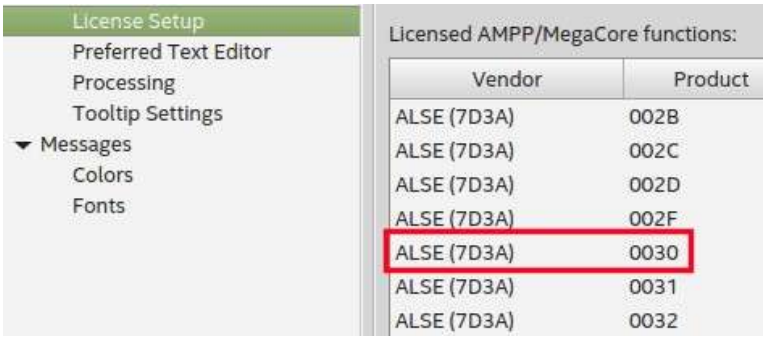
The following pages will guide you through the ALSE's Aurora 8b10b Hardware Tester Reference design **for clients having purchased the ALSE Aurora 8B/10P IP for Intel FPGAs.**

Here are the **steps** to follow :

- Open the provided ALSE Quartus project, located typically in *./fit/altera/board_name/*
- Check (and adjust if the delivered source files allow it) the Aurora IP configuration defined in the Top Level VHDL generics :
  - Device Family
  - Number of XCVR Lanes
  - Lane Width (16 or 32)
  - Streaming / Framing Mode
  - CRC Enabled (if using Framing)
  - Duplex or Simplex Tx/Rx Mode
  - etc …

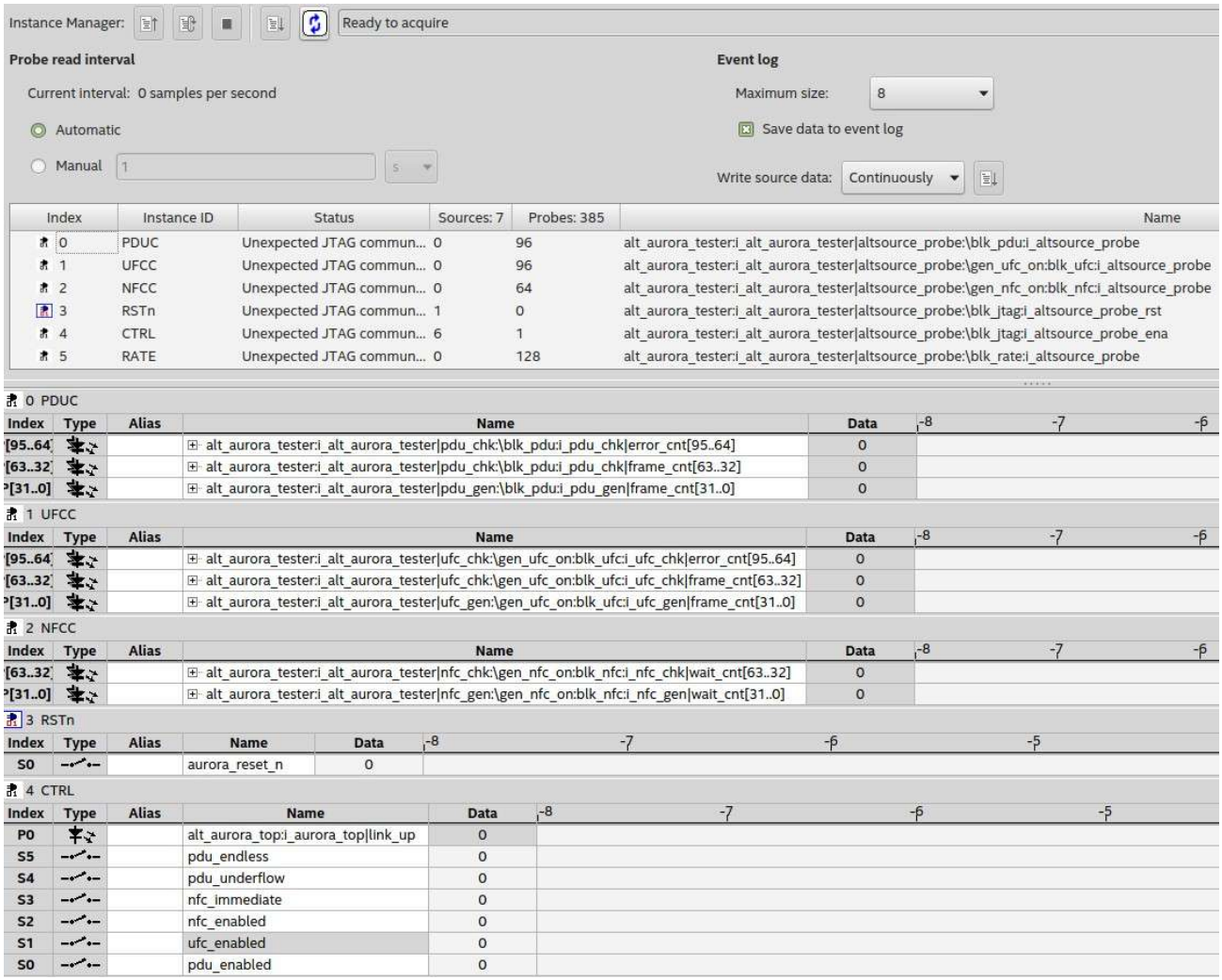
```
16 ---------------------------------------
17 entity avdb_x1_top_streaming is
18 ---------------------------------------
19
20   generic (
21     device_family  : string    := "CycloneV"; -- Altera Device Family
22     nb_lanes       : integer   :=  1;         -- Number of XCVR lanes
23     lane_width     : integer   :=  32;        -- 16 or 32
24     streaming_mode : std_logic := '1';        -- Streaming Mode if '1', Framing if '0'
25     crc_enable     : std_logic := '0';        -- CRC Enabled if '1', Disabled if '0'
26     simplex_tx     : std_logic := '0';        -- Simplex Tx only if '1'
27     ext_clk_freq   : positive  := 50e6        -- Ext Clock Frequency
28   );
```

- Check the pinout assignments in the project (through the Pin Planner, or directly in the QSF)
  The Hardware Tester Reference design uses typically the following pins :
  - an external free-running clock (board quartz) called *ext_clk*. This clock frequency should match the *ext_clk_freq* VHDL generic (example : ext_clk_freq = 50e6 if the *ext_clk* pin runs @ 50MHz)
  - the XCVR Tx/Rx pins (number depends on the *nb_lanes* generic) and the XCVR Reference Clock. These pins might be associated with an SFP(+) cage, QSFP(+) connector, or to a specific Board implementation.
- If using the encrypted version of the IP, check that you have the license installed correctly : in Quartus, go to the Tools > License Setup... menu, and check that the ALSE (7D3A Vendor ID) **0030** Product (this is the Aurora 8b10b IP) appears in the list, with a valid expiration date.

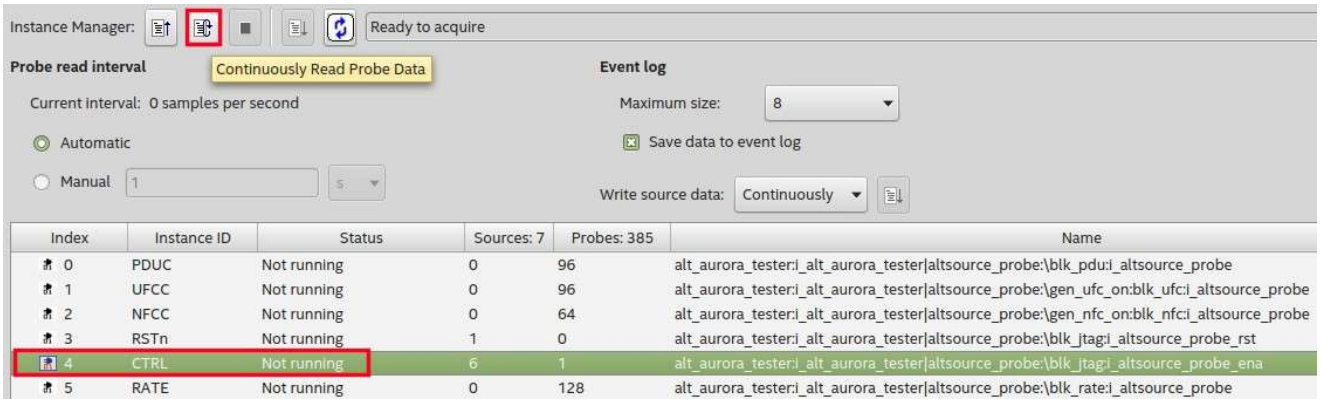


- Once this is OK, compile the project (**CTRL+L** in Quartus).
  The project should compile without any error, and no timing violation should be reported.
- Prepare your hardware XCVR connections/cables.
  Connect it to the Polarfire Dev kit implementing the simple loopback on PU/UFC/NFC packets design.
- Power on your board(s), plug the JTAG (USB Blaster typically), and program the board with the generated bitstreams (.SOF file for Intel). **Polarfire should be programmed as well with ALSE loopback bitstream !**
- Back in Quartus, open the provided ***Tools > In-System Sources and Probes Editor*** file :
  File > Open, and select the *./fit/altera/board_name/**insys_src_prb.spf*** file.
  A window like the one below will appear.

  Note : if the Aurora Streaming IP is used, the UFCC and NFCC instances will not be present.



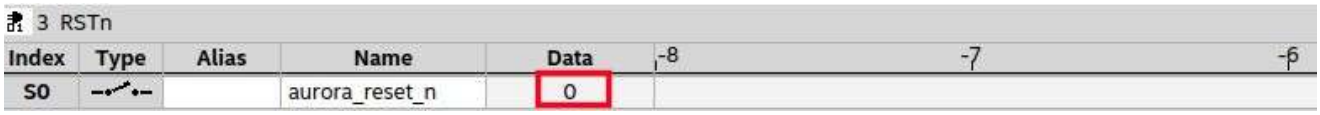
- First, Check in the JTAG configuration window that the correct JTAG cable is selected.
- Select the ***CTRL*** instance, and click on *Continuously Read Probe Data*.



Index	Instance ID	Status	Sources: 7	Probes: 385	Name
0	PDUC	Not running	0	96	alt_aurora_tester:i_alt_aurora_tester|altsource_probe:\blk_pdu:i_altsource_probe
1	UFCC	Not running	0	96	alt_aurora_tester:i_alt_aurora_tester|altsource_probe:\gen_ufc_on:blk_ufc:i_altsource_probe
2	NFCC	Not running	0	64	alt_aurora_tester:i_alt_aurora_tester|altsource_probe:\gen_nfc_on:blk_nfc:i_altsource_probe
3	RSTn	Not running	1	0	alt_aurora_tester:i_alt_aurora_tester|altsource_probe:\blk_jtag:i_altsource_probe_rst
4	CTRL	Not running	6	1	alt_aurora_tester:i_alt_aurora_tester|altsource_probe:\blk_jtag:i_altsource_probe_ena
5	RATE	Not running	0	128	alt_aurora_tester:i_alt_aurora_tester|altsource_probe:\blk_rate:i_altsource_probe

You will see '0's being read for all bits of the *CTRL* instance, including for the **link_up** bit.

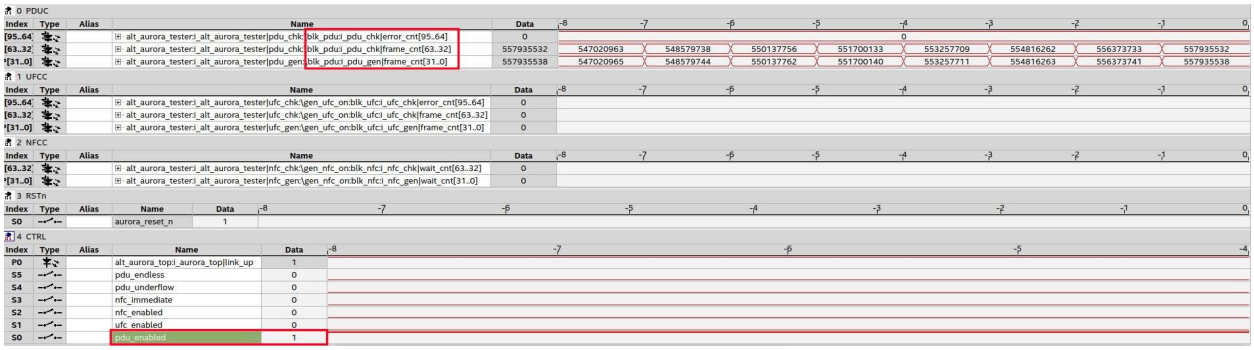
- To start the test, you must first release the Aurora Tester (and IP) **active_low reset**, which is asserted by default ('0') at FPGA start-up. This can be done by clicking on the '0' value of the *Data* Column of the **RSTn** instance, making the '0' value change into a '1' => the reset is removed (de-asserted).



The **link_up bit of the CTRL instance should be now on** ('1'), signifying that the Aurora link is up through the full setup (board-to-board, or Hardware Loopback through a connector).

<u>Note</u> : If link is not up, check the pinout assignments and the XCVR Ref Clock frequency.
Check also that the Slave board (if any) runs at the same link rate than the Intel board.
If any error is found, correct it and recompile.
If the link is still down, a SignalTap core is embedded in the FPGA to help you debug some of the signals that may help you understand why. Please contact ALSE to help you.

- **<u>To start testing PDU frames</u>**, first enable **<u>Continuous Read</u>** also on the **PDUC** instance.
  Then, in the *CTRL* instance, assert ('1') **pdu_enabled** (bit S0).



You should see in the PDUC instance both PDU_GEN / PDU_CHK Frame Counters increase quickly, and the PDU_CHK Errors Counter stay at 0, meaning PDU Frames are sent/received without any error.

- PDU tests can be adjusted by enabling :
  - the **pdu_underflow** bit (*CTRL* instance, bit S4) : if asserted, the PDU Frame generator insert dead cycles (no tx_pdu_vld asserted) at random clock cycles, thus decreasing a bit the PDU Bandwidth.
  - the **pdu_endless** bit (*CTRL* instance, bit S5) : if asserted, the PDU Frame generator creates a frame without EOP (Endless “infinite” frame) : the Frame Counter will stay to its current value, as long as this bit is asserted.

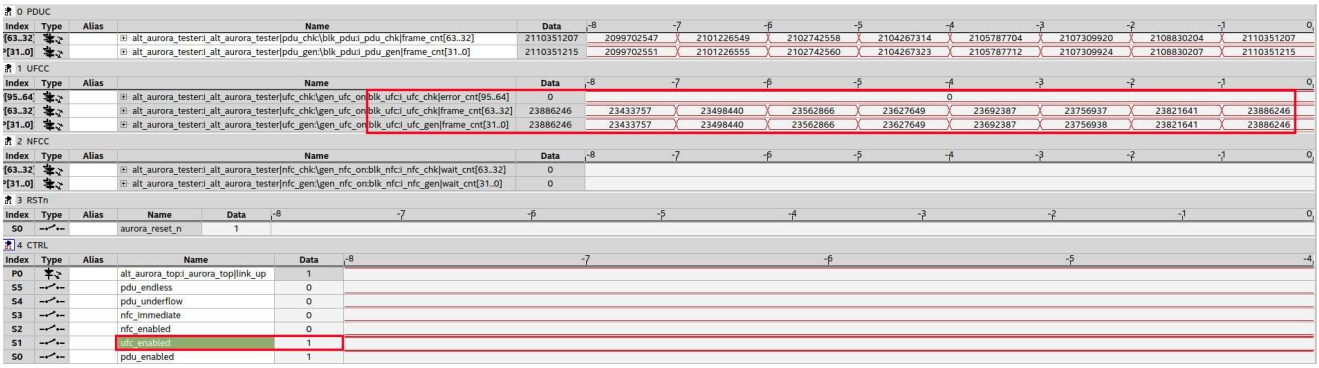
- PDU tests can also be driven by a **Fixed mode** : this mode generates known PDU Frames sizes, with sizes and interframe gaps adjustable dynamically.
  The data generated are again incremented (“counter”) values.

  To use this mode :
  - First stop the PDU Stream (pdu_enabled = ‘0’).
  - Then, set the **pdu_fixed_size_on** bit (PDUF instance, bit S0)
  - Set the maximum/minimum frame sizes :
    pdu_fixed_size_max and **pdu_fixed_size_min** (should be at least 4 minimum).
  - Set the frame size increment : **pdu_fixed_size_incr** (0 means frame size is always the same, max at start-up, 1 means after each frame finished, the next frame has a size +1, etc …)
  - Set the Inter-Frame gap, in clock cycles : **pdu_fixed_size_ifgap** : should be at least 1

Optionally :

  - Set the frame cut size : **pdu_fixed_size_cut_size** : this is the number of data valid after which a gap will be inserted in the data_valid (in order to avoid PDU Frames being a continuous burst).
  - Set the frame cut gap : **pdu_fixed_size_cut_gap** defines the number of clock cycles of the “cut”.
  - **Enable** the PDU Stream again (pdu_enabled = ‘1’)

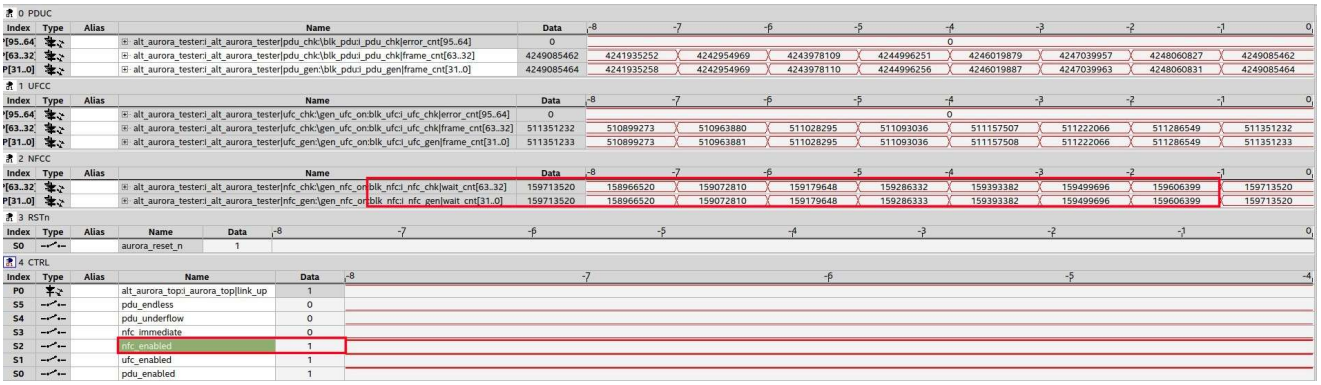
- **To start testing UFC frames**, first enable Continuous Read also on the **UFCC** instance.
  Note : this is only possible in **Framing** mode.
- Then, in the *CTRL* instance, assert (‘1’) **ufc_enabled** (bit S1).



As for PDU, you should see in the UFCC instance both UFC_GEN / UFC_CHK Frame Counters increase quickly, and the UFC_CHK Errors Counter stay at 0, meaning UFC Frames are sent/received without any error.

Note that you can enable both PDU and UFC testing at the same time.

- **To start testing NFC frames**, first enable Continuous Read also on the **NFCC** instance.
  Note : this is only possible in **Framing** mode.
- Then, in the *CTRL* instance, assert (‘1’) **nfc_enabled** (bit S2).

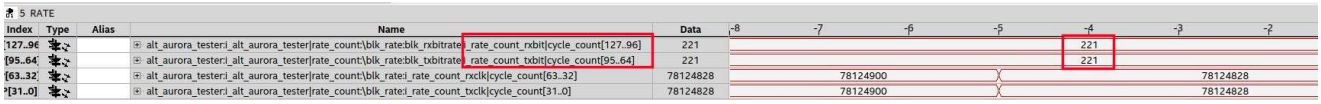


*You should see in the NFCC instance both NFC_GEN / NFC_CHK **Wait** Counters increase quickly with the same value, meaning NFC Frames are sent/received, without any error.*

Note that you can enable all PDU, UFC, and NFC testing at the same time.

- NFC tests can be adjusted by enabling the **nfc_immediate** bit (*CTRL* instance, bit S3) : if asserted, the NFC Frames are sent in Immediate mode, otherwise (‘0’, default), the NFC **Completion** mode is used.

- Finally, the *In-System Sources and Probes Editor* contains a **RATE** instance, allowing to observe in real-time the PDU Bandwidth exactly Transmitted (Tx) and Received (Rx), in MB/s (MegaBytes/second).
  You can perform a **Continuous Read** also on this instance, during PDU/UFC/NFC testing.



3.4.2. Xilinx Master Design

The Xilinx Master Design offers the same features as Intel design.

The same steps should be followed to generate the design.

Here we list only the differences. Refer to previous chapter for more detailed information.

- Open the provided ALSE Xilinx project, located typically in *./fit/xilinx/board_name/*.
  It is intended for Vivado 2020.2 and could require some IP update with newer version.
- Check (and adjust if the delivered source files allow it) the Aurora IP configuration defined in the Top Level VHDL generics :
  - Number of Transceivers Lanes
  - Lane Width (16 or 32)
  - Streaming / Framing Mode
  - CRC Enabled (if using Framing)

  The default configuration should indeed match the Polarfire settings !
- Check the pinout assignments in the project **(directly in the .xdc file)**.
  The Hardware Tester Reference design uses typically the following pins :
  - An external free-running clock (board quartz) called *ext_clk* used to generate a POR and a 100 MHz system with a PLL.
  - The XCVR (Transceiver) Tx/Rx pins (number depends on the *nb_lanes* generic) and the XCVR Reference Clock.
    These pins might be associated to an SFP(+) cage, QSFP(+) connector, or to a specific Board implementation.
  - User LEDs used to display basic debug information (link-up, reset...)
- Compile the project !
- Prepare your hardware Connections and Cables.
  Connect them to the Polarfire development kit.
  Verify that the Polarfire kit is programmed with the simple loopback on PU/UFC/NFC packets.
- Verify that the programming cable is correctly connected.
- Power On your boards, and if needed program the Master board with the adhoc bitstream (.BIT file for Xilinx). Reminder : the **Polarfire should be programmed as well with ALSE loopback bitstream !**

- Once your board is programmed you should see the VIO entity in the Vivado Hardware Manager. Here is the list of the source and probe connection.

  For more information on a specific signal, refer to the Intel section !

Direction	Index	Size	Signal Name
OUT	0	1	pdu_tx_ena
OUT	1	1	pdu_underflow
OUT	2	1	nfc_tx_ena
OUT	3	1	ufc_tx_ena
OUT	4	1	pdu_fixed_size_on
OUT	5	32	pdu_endless_mode (bit 0 only is used)
IN	0	32	pdu_rx_frame_cnt
IN	1	32	pdu_rx_error_cnt
IN	2	32	ufc_rx_frame_cnt
IN	3	32	ufc_rx_error_cnt
IN	4	32	pdu_tx_frame_size
IN	5	32	pdu_tx_frame_cnt
INOUT	6	32	pdu_fixed_size_max
INOUT	7	32	pdue_fixed_size_min
INOUT	8	32	pdu_fixed_size_incr
INOUT	9	32	pdu_fixed_size_ifgap
INOUT	10	32	pdu_fixed_cut_size
INOUT	11	32	pdu_fixed_cut_gap

Table : Xilinx sources and probes

3.5. Pictures

The picture below shows boards interconnect in a **x1 configuration** (using SFP+ connectors + optical fiber cable), between the ReflexCES **CycloneV-based Clovis board** (also known as the ALSE “AVDB” Board), on the bottom of the picture, and the Microsemi **Polarfire MPF300 evaluation development kit (MPF300-EVAL-KIT)** (on the top).

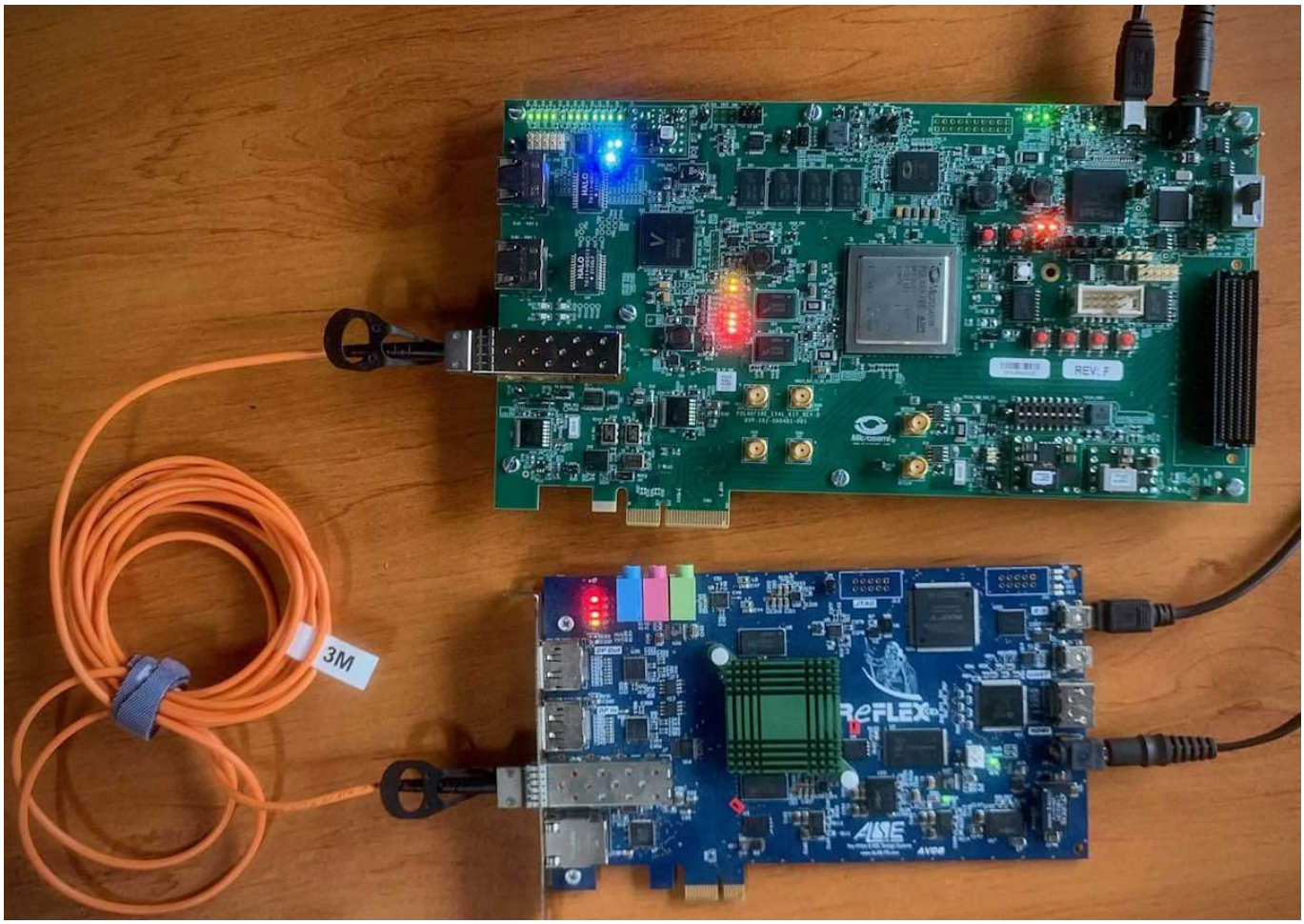


Figure 3 : x1 lane hardware test bench overview

The picture below shows the state of the **Polarfire MPF300 Development kit** board once programmed and not connected to any master board.

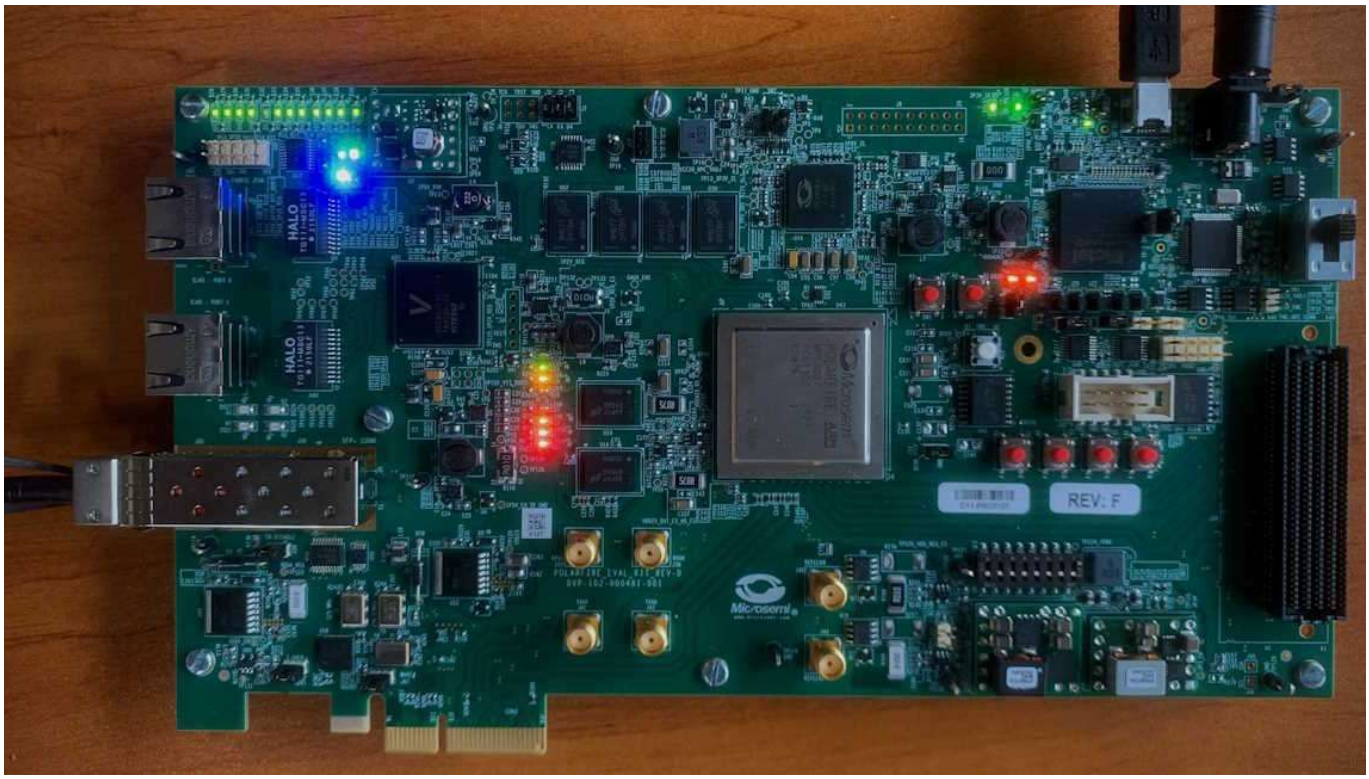


Figure 4 : Polarfire MPF300 Development Kit

3. TECHNICAL SUPPORT

For any type question about this IP, please contact ALSE Technical Support by **E-mail** at support@alse-fr.com or at a specific E-mail address that you may have received.

If a telephone contact is desired, our R&D office can be reached at +33 1 84 16 32 32, during office hours, CET.

You will be either answered directly or directed to an engineer available and competent, depending on the type of question or support.

ALSE provides a warranty defined In the IP Licensing Agreement, which is strictly limited to making the best efforts in fixing any design error that would prevent the normal use of the IP or induce an incorrect behavior of the IP, as long as the IP is used in a correct and valid context. This warranty will be void if any modification is made to the code by the customer.

This IP can only be purchased and used after signing the ***ALSE IP License Agreement***.

-=oOo=-