

© 2023 - [ALSE](https://alse-fr.com)

Date : Novembre 2023

Version : 2023.12c



Advanced Logic Synthesis for Electronics
A.L.S.E. - <https://alse-fr.com>

ALSE's 10GEDEK 10G Ethernet IP for Polarfire Reference Design User Guide

To contact ALSE, visit the
[ALSE website \(FPGA.fr\)](https://fpga.fr)
or write to : info@alse-fr.com

Table of Contents

I - Document history.....	2
II - Archive.....	2
III - Introduction.....	3
The 10GEDEK IP.....	3
The hardware platform.....	4
IV - Preparation.....	5
Programming the Polarfire Kit.....	5
Preparing the setup.....	5
Preparing the 10G Ethernet access.....	5
Preparing the PC – Software compilation.....	5
V - Testing the Reference Design.....	6
Power up the FPGA Kit.....	6
Testing from the PC.....	6
First step : PING.....	6
Using the Reference Design Software.....	7
Principle.....	7
FPGA RX tests.....	9
FPGA TX tests.....	10
VI - Conclusion.....	11

I - Document history

Version	Date	Description
v2023.11a	Nov 2023	Initial Version
v2023.12a	Dec 2023	Complete version
v2023.12b & c	Dec 2023	Minor revisions

II - Archive

This document is part of an archive which you can **download from this link**, containing :

ALSE_10GEDEK_Polarfire.pdf	This document
10GEDEK_mpf300_EvalKit.ppd	Binary Programming file for Polarfire Eval kit.
soft/*	Reference Design test Software (multi-platforms, to be compiled) Use the provided Makefile script.
T:\Syndocs\IPs\10GEDEK\ Polarfire\soft\gedek_api\win64\ ref_design64bit.exe	Reference design software, compiled for Windows 64 bits. Note : the exe needs gedek_api.dll (put it in the same directory).

III - Introduction

At ALSE, we are grateful to Microchip in Germany : they had a Polarfire kit available and they loaned it to us for a few weeks, thus allowing us to port and test several of our IPs :

- ✓ GEDEK (Gigabit Ethernet processor-less IP)
- ✓ 10GEDEK (10G Ethernet processor-less IP)
- ✓ Aurora 8B/10B
- ✓ Aurora 64B/66B

If you are interested by one of the above IPs, please [contact ALSE](#).

As seen above, we have ported our [10GEDEK](#) IP to the Microchip Polarfire family and specifically to the Polarfire Development kit. If you have this kit, you will be able to test our 10GEDEK Reference Design ! This is demonstrated in this step-by-step User Guide. If you are not familiar with 10GEDEK, follow the link above for a description.

The complete design uses a small fraction of the FPGA (less than 5%) and provides the complete bi-directional 10 Gigabits Ethernet communication, capable of loss-less transfer of up to twice 1.12 Gbytes/s (in either direction), simultaneously.

The 10GEDEK IP

The complete Reference Design including the 10GEDEK IP occupies less than 5% of the Polarfire.

Here are the details :

Module Name	Fabric 4LUT	Fabric DFF	Interface 4LUT	Interface DFF	Single-Ended I/O	uSRAM (64x12)	LSRAM (20K)
Top	12972	10677	2196	2196	17	18	53
Primitives	31	11	0	0	17	0	0
FREQ_METER161_i	119	88	72	72	0	0	0
gedek10G_phy_top_microsemi_inst	1639	2669	144	144	0	12	0
i0_GEDEK10G_REF_DESIGN_TOP	10981	7813	1980	1980	0	6	53
Primitives	63	190	0	0			
i_ETH_MUX	92	75	0	0			
i_ETH_SEND_CNT	655	360	0	0			
i_ETH_STREAM_GEN_CHK	639	330	0	0			
i_GEDEK_10G	9532	6858	1980	1980			

Resource Usage

Type	Used	Total	Percentage
4LUT	15168	299544	5.06
DFF	12873	299544	4.30
I/O Register	0	1536	0.00
User I/O	17	512	3.32
-- Single-ended I/O	17	512	3.32
-- Differential I/O Pairs	0	256	0.00
uSRAM	18	2772	0.65
LSRAM	53	952	5.57
Math	2	924	0.22
H-Chip Global	5	48	10.42
Local Global	2	1008	0.20
PLL	1	8	12.50
DLL	0	8	0.00
CRN_INT	1	24	4.17
INIT	1	1	100.00
OSC_RC160MHZ	1	1	100.00
Transceiver Lanes	1	16	6.25
Transceiver PCIe	0	2	0.00
TX_PLL	1	11	9.09
XCVR_REF_CLK	1	11	9.09
ICB_CLKDIV	1	24	4.17
ICB_CLKINT	2	72	2.78

As we can see, the 10GEDEK IP is very compact and fits easily in a small fraction of the FPGA, even associated with the Reference Design features, data generators and checkers, register interface etc.

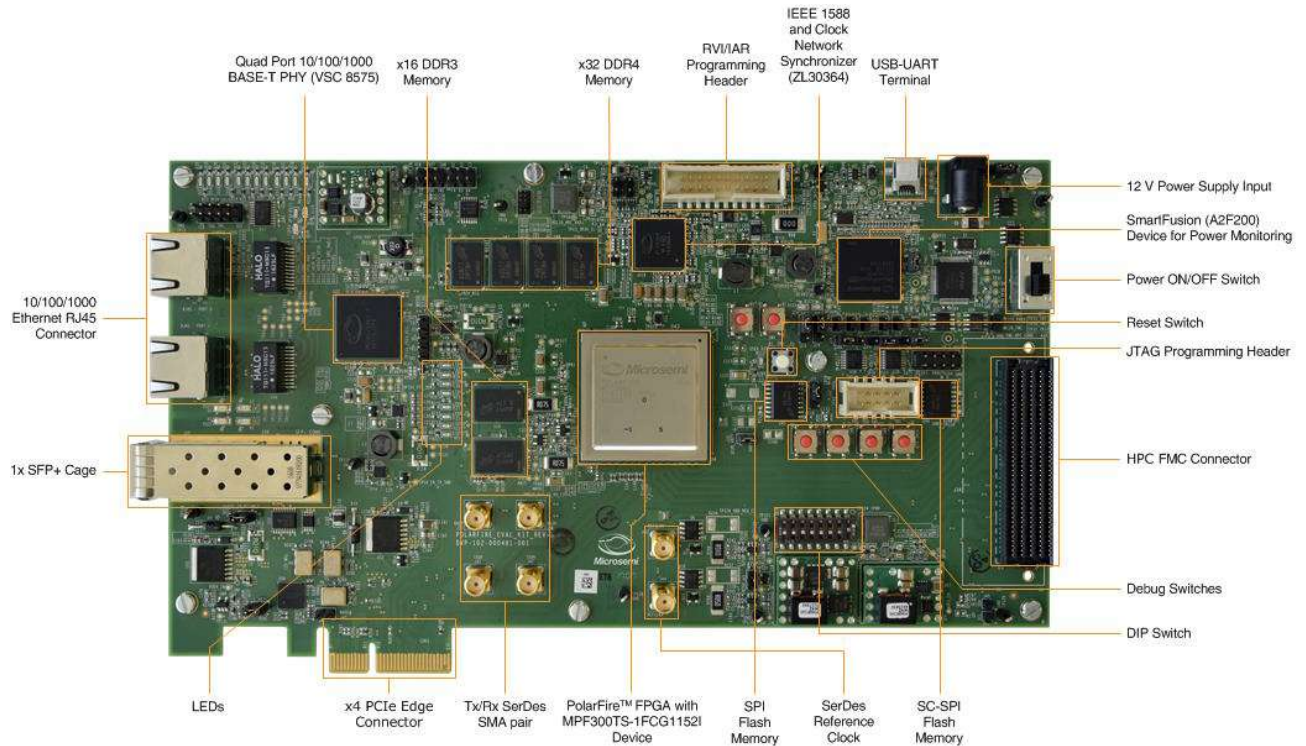
As we will see, the PC side is also straightforward and does not necessitate complex hardware nor software.

The hardware platform

To test the IP, we use the **Polarfire FPGA Evaluation kit (MPF300-EVAL_KIT)**.

This board has an SFP+ cage inside which we can insert a 10G Direct Attach Copper cable or a 10G SFP+ optical cable connected on the other end to a PC's 10G Ethernet card or to a 10G switch.

The Libero Free License (Silver) does support the FPGA used in this kit. You don't have to invest in the paid-mode license to compile designs on this Kit.



Note that we also used this platform to port and test our **Aurora IPs** (8b/10b and 64b/66b).

By using the same SFP connector, but with a 1Gb RJ45 adapter module, we also tested our **GEDEK IP**.

We chose to not port and test our JEDS204B Tx + Rx IPs since Microchip offers a solution and we had a limited amount of time dedicated to this platform.

IV - Preparation

Here are the steps to prepare the Polarfire kit and the PC.

Programming the Polarfire Kit

You should program the provided programming file (**10GEDEK_mpf300_EvalKit.ppd**) in the Polarfire FPGA using FPexpress.

Preparing the setup

Apart from the kit, you will need at least :

- A **PC** with a **10G Ethernet SFP+ card** in it (which can be found for about \$50). Both Windows and Linux will be suitable.
- Ideally, you need a **C compilation platform**. Linux users will certainly have this already. To make it simple to Windows users, we provide a Windows 64 bits executable, ready to use.
- A proper **10G SFP+ Cable**, either :
 - DAC (Direct Attach Copper cable), 1 to 2 meters, cost between \$9 and \$20.
 - or :
 - Optical Cable (SFP+ Active Fiber cable). More expensive but can reach tens of meters (or more).

Use the cable to connect the Polarfire kit to the PC 10G Ethernet card.

Note : you can use a 10G switch. Our IP works through any network that enables UDP traffic. In this design we use the Ports = 1234,1235 and 1237 (decimal).

Preparing the 10G Ethernet access

In the design used for this demo, our 10GEDEK IP inside the **FPGA** is configured in its simple mode, with fixed a network **IP address** = **192.168.5.32**. and **MAC** = 00-07-ED-A8-01-20.

You must configure your PC 10G card properties, to create the proper local network like :

- ◆ **Manual IP Address**
- ◆ IP Address = **192.168.5.1**
- ◆ Network **Mask** = 255.255.255.0

Preparing the PC – Software compilation

On the PC, you need to compile the Reference Design testing Software. It is located under the **soft/** folder using the **makefile** script provided.

Open the **makefile** script with a text editor. Depending on the Operating System (Linux / Windows, 32 or 64 bits), select (**un-comment**) the suitable compile command :

```
# Linux | Cygwin :
# export CROSS_COMPILE=                && make clean    && make

# Linux Mingw Cross 32 :
# export CROSS_COMPILE=i686-w64-mingw32-  && make clean    && make

# Linux Mingw Cross 64 :
# export CROSS_COMPILE=x86_64-w64-mingw32- && make clean    && make

# Windows 32 Native Mingw :
# set ARCH=win32&& mingw32-make clean && mingw32-make

# Windows 64 Native Mingw :
# set ARCH=win64&& mingw32-make clean && mingw32-make
```

V - Testing the Reference Design

Power up the FPGA Kit

- Make sure the PC is up and running and prepared as described in the previous chapter.
- Make sure the proper 10G SFP+ cable is connecting the FPGA kit and the PC.
- Make sure the FPGA kit is correctly programmed with the provided bitstream.
- Power up the FPGA board : you **must** plug **both** the 12V Power Supply **and** the JTAG usb cable ! This is explained in the kit's Manual.
 - ◆ **LED 4** should display a heartbeat, attesting that the 180 MHz *system* clock is operational. This clock is generated from the external 50 MHz clock (pin E25).
 - ◆ **LED 5** : another heartbeat, based on the 156.25 MHz Transceiver Reference Clock.
 - ◆ **LED 6** : heartbeat if the 166.133 MHz clock is recovered from the 10G Ethernet Rx link.
 - ◆ **LED 7** should be OFF if the 10G Ethernet Link is operational. If LED 7 is ON, the link is not established.
 - ◆ All other LEDs should be OFF.

Testing from the PC

Verify that LED 6 beats and LED 7 is off.

First step : PING

Start by **pinging** the board to check the connectivity :

```
ping 192.168.5.32
```

If it doesn't succeed :

- ✓ Double check that all the preparation steps have been performed.
- ✓ Verify the status on the FPGA board LEDs. LED7 should be OFF
- ✓ Check also again the Ethernet settings for the PC 10G board (IP=192.168.5.1).



Once pinging is successful, you can proceed with the actual tests (next pages).

Make sure you have a compiled version of the reference Design Software for your platform.

If on Windows (10), you can use the provided executable. Make sure you have the proper gedek_api.dll located in the same directory.

Using the Reference Design Software

Launch the Reference Design software.

It will start by pinging the board (twice) to establish the connection. Doing so actually updates the remote device IP address inside the FPGA ! This allows the FPGA to know where to send test data to.

```
] Gedek Reference Design Version is v2023.11.25

> Note, this compiler uses sizeof(unsigned long) = 8

> Using IP Address 192.168.5.32 !
PING 192.168.5.32 (192.168.5.32) 64(92) bytes of data.
72 bytes from 192.168.5.32: icmp_seq=1 ttl=128 time=0.103 ms
72 bytes from 192.168.5.32: icmp_seq=2 ttl=128 time=0.105 ms

--- 192.168.5.32 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1024ms
rtt min/avg/max/mdev = 0.103/0.104/0.105/0.001 ms

> Threads Launched ! [CTRL+C to Exit]

] Reading FPGA VERSION ...

FPGA TX Stream is NOT enabled by default : press BUTTON 0, 1 or 2 to enable it !
FPGA TX Stream can be alternatively enabled by using the interface register

FPGA RXCNT Stream registers are located @0x20 (bit 0 = Start/Stop) W 20 1 / W 20 0
> FPGA Version is : 2023.11.28

> What do you want to do (TX PAYLOAD_SIZE = 8192) ?
1 = Send One Counter Data Frame to the FPGA
e = Send One Counter Data Frame to the FPGA with bad header !
d = Send One Counter Data Frame to the FPGA with bad counter !
5 = Send Counter Data during 5 secs to FPGA
R = Read followed by Address (e.g : r 0)
W = Write followed by Address + Data (e.g : w 2 a75e)
0 = Stop Transmission from the FPGA
Q = Quit Program
```

Read the FPGA bitstream version (Register @ addresss 0):

```
R 0
```

You should get the same information as in the initialization information reproduced above.

Note : during the tests, an interesting register to read during the tests is the RX error register located at address 6, which counts the errors detected on the frames received by the FPGA.

```
R 6
```

We are now ready to exercise the link and stress the PC and the network !

Principle

Both the PC Software and the FPGA design have the ability to emit and receive Ethernet UDP frames containing incrementing 32 bits data words (32 bits counter).

To maximize the performance on the PC side (and minimize the protocol overhead), we have chosen to **enable the Jumbo frames** option of our IP. At 10G, the larger payload per frame is helping the PC to cope with the received bandwidth, and also to emit more data per second.

FYI : in this design, we use the **UDP ports** number 1234, 1235 and **1237** (dec).

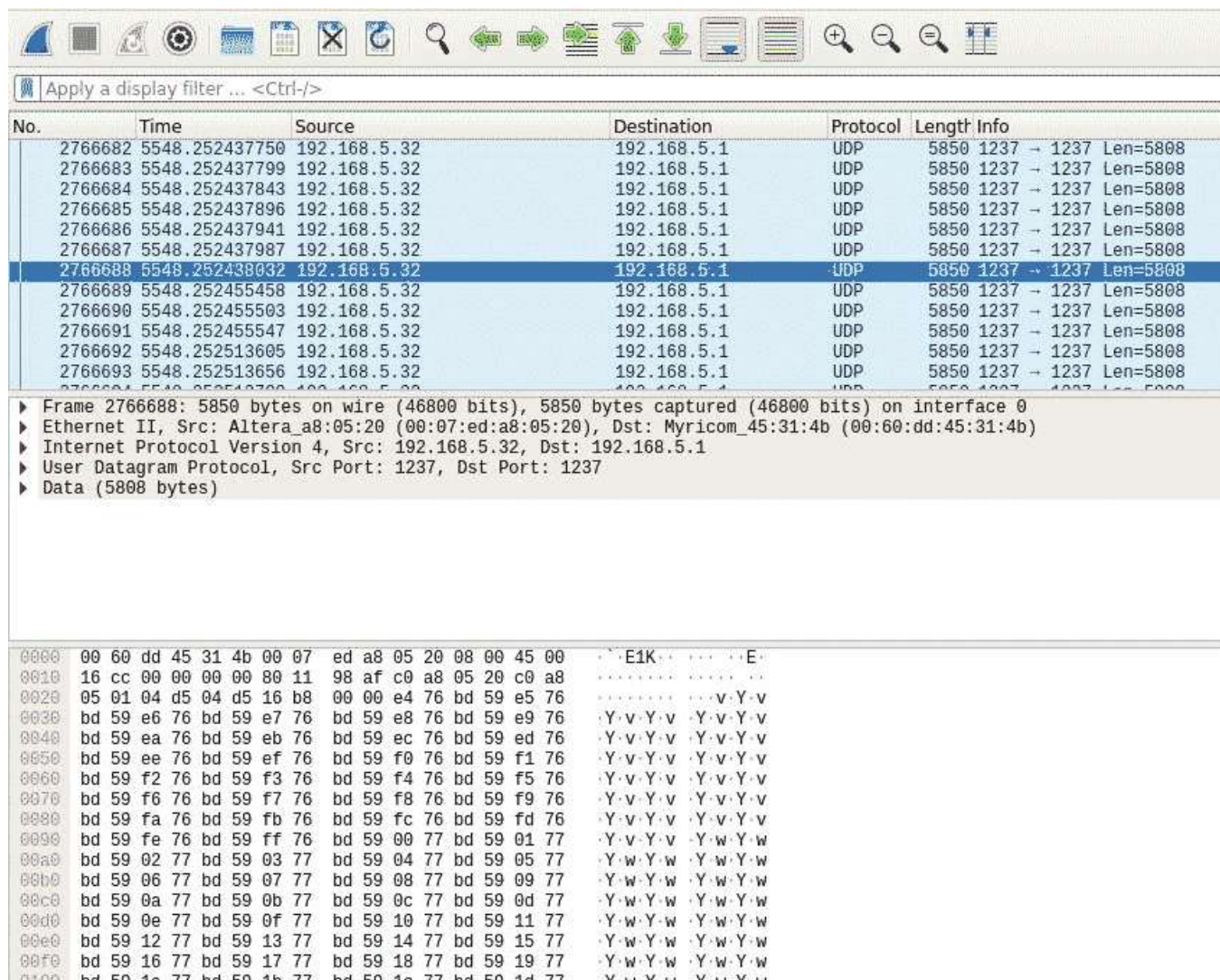
The first 32 bits of the Ethernet frame payload is a **frame index**. Both the FPGA design and the Ref Design Software check that this value is continuous when receiving frames : this demonstrates that no frame is lost or misplaced. An error counter does count up when the Frame Index is different from the FI from the previous frame + 1.

The rest of the payload is a continuous counter.

The FPGA design does also check the entire payload.

The PC software does **not** verify the entire payload : it would be a lot of stress on the PC's processor !

Here is a frame received by the PC and captured using Wireshark :



No.	Time	Source	Destination	Protocol	Length	Info
2766682	5548.252437750	192.168.5.32	192.168.5.1	UDP	5850	1237 → 1237 Len=5808
2766683	5548.252437799	192.168.5.32	192.168.5.1	UDP	5850	1237 → 1237 Len=5808
2766684	5548.252437843	192.168.5.32	192.168.5.1	UDP	5850	1237 → 1237 Len=5808
2766685	5548.252437896	192.168.5.32	192.168.5.1	UDP	5850	1237 → 1237 Len=5808
2766686	5548.252437941	192.168.5.32	192.168.5.1	UDP	5850	1237 → 1237 Len=5808
2766687	5548.252437987	192.168.5.32	192.168.5.1	UDP	5850	1237 → 1237 Len=5808
2766688	5548.252438032	192.168.5.32	192.168.5.1	UDP	5850	1237 → 1237 Len=5808
2766689	5548.252455458	192.168.5.32	192.168.5.1	UDP	5850	1237 → 1237 Len=5808
2766690	5548.252455503	192.168.5.32	192.168.5.1	UDP	5850	1237 → 1237 Len=5808
2766691	5548.252455547	192.168.5.32	192.168.5.1	UDP	5850	1237 → 1237 Len=5808
2766692	5548.252513605	192.168.5.32	192.168.5.1	UDP	5850	1237 → 1237 Len=5808
2766693	5548.252513656	192.168.5.32	192.168.5.1	UDP	5850	1237 → 1237 Len=5808

▶ Frame 2766688: 5850 bytes on wire (46800 bits), 5850 bytes captured (46800 bits) on interface 0
 ▶ Ethernet II, Src: Altera_a8:05:20 (00:07:ed:a8:05:20), Dst: Myricom_45:31:4b (00:60:dd:45:31:4b)
 ▶ Internet Protocol Version 4, Src: 192.168.5.32, Dst: 192.168.5.1
 ▶ User Datagram Protocol, Src Port: 1237, Dst Port: 1237
 ▶ Data (5808 bytes)

0000	00 60 dd 45 31 4b 00 07	ed a8 05 20 08 00 45 00	E1K
0010	16 cc 00 00 00 00 80 11	98 af c0 a8 05 20 c0 a8	
0020	05 01 04 d5 04 d5 16 b8	00 00 e4 76 bd 59 e5 76	v.Y.v
0030	bd 59 e6 76 bd 59 e7 76	bd 59 e8 76 bd 59 e9 76	Y.v.Y.v Y.v.Y.v
0040	bd 59 ea 76 bd 59 eb 76	bd 59 ec 76 bd 59 ed 76	Y.v.Y.v Y.v.Y.v
0050	bd 59 ee 76 bd 59 ef 76	bd 59 f0 76 bd 59 f1 76	Y.v.Y.v Y.v.Y.v
0060	bd 59 f2 76 bd 59 f3 76	bd 59 f4 76 bd 59 f5 76	Y.v.Y.v Y.v.Y.v
0070	bd 59 f6 76 bd 59 f7 76	bd 59 f8 76 bd 59 f9 76	Y.v.Y.v Y.v.Y.v
0080	bd 59 fa 76 bd 59 fb 76	bd 59 fc 76 bd 59 fd 76	Y.v.Y.v Y.v.Y.v
0090	bd 59 fe 76 bd 59 ff 76	bd 59 00 77 bd 59 01 77	Y.v.Y.v Y.w.Y.w
00a0	bd 59 02 77 bd 59 03 77	bd 59 04 77 bd 59 05 77	Y.w.Y.w Y.w.Y.w
00b0	bd 59 06 77 bd 59 07 77	bd 59 08 77 bd 59 09 77	Y.w.Y.w Y.w.Y.w
00c0	bd 59 0a 77 bd 59 0b 77	bd 59 0c 77 bd 59 0d 77	Y.w.Y.w Y.w.Y.w
00d0	bd 59 0e 77 bd 59 0f 77	bd 59 10 77 bd 59 11 77	Y.w.Y.w Y.w.Y.w
00e0	bd 59 12 77 bd 59 13 77	bd 59 14 77 bd 59 15 77	Y.w.Y.w Y.w.Y.w
00f0	bd 59 16 77 bd 59 17 77	bd 59 18 77 bd 59 19 77	Y.w.Y.w Y.w.Y.w
0100	bd 59 1a 77 bd 59 1b 77	bd 59 1c 77 bd 59 1d 77	Y.w.Y.w Y.w.Y.w

Note : this was captured during a test with a payload (5808) smaller than 8192 ("smaller Jumbo").

Important if you are interested by the exact frame contents :

- The payload of the Frames *generated by the FPGA* start with the Frame Index Word, followed by 2047 a progressive words (counter). This progressive value is not restarting between frames ! This means that all the payloads are different between frames. The progressive value is used by the comparison with register 0x22 (see later).
- The payload of the Frames generated by the PC software start with the incrementing Frame Index Word, but are followed by the same values (incrementing from 0). Why ? Because it would be difficult for the PC to calculate a new payload for every frame sent, so the software only changes the frame Index in the buffer before each send frame call.

FPGA RX tests

To test the reception of the 10GEDEK IP, you can send frames to the FPGA through the Reference Design Software. The software sends frames as fast as possible !

You must understand that you are testing *the performance of the PC* and the fact that the FPGA does receive correct frames.

But the FPGA Rx performance is 100% of theoretical bandwidth !

10GEDEK can receive back-to-back frames with zero loss.

Type :

```
5 <Enter>
```

to send frames continuously during 5 seconds, as fast as possible for the PC.

LED 9 should be ON during these 5 seconds, indicating good data reception on the FPGA side.

The bandwidth is reported in the shell.

Ignore the first batch (which is not complete) and the last batch.

```
5
Send for 5 seconds as fast as possible to the remote board
TX Speed = 532968 Ko/s
TX Speed = 1208144 Ko/s
TX Speed = 1208992 Ko/s
TX Speed = 1205328 Ko/s
TX Speed = 1189672 Ko/s
```

We see that the PC was able to send **1.2 Giga (octets) bytes per second**, i.e. **9.6 GigaBits/s**.

This is the maximum theoretical throughput when using Jumbo frames : the frames are sent back-to-back.

If you see lower values, it means your PC isn't fast enough to send that much data through the 10G Ethernet card.

Check the error register (0x6) for reception errors.

A couple of errors could be normal (due to the initialization of the massive test).

In case of error, **LED 11** should blink !

You can generate intentional errors using options "e" or "d" :

```
e
Sending 1 Counter Data frame to the FPGA
```

Check the error register again :

```
R 6
> Register @0x00000006 => 0x00000001
```

You can also test sending an error on the payload (command "d").

FPGA TX tests

For this test, the FPGA will send Jumbo frames to the PC, up to the maximum theoretical speed (back-to-back frames). The FPGA emission is controlled through Push Buttons when available, or (as for this Polarfire Reference Design), through **32 bits FPGA registers** described below, which can be accessed through Ethernet.

You must understand that you are again testing *the performance of the PC : its ability to receive a lot of frames without loss*.

The FPGA Tx performance is also 100% of theoretical bandwidth ! 10GEDEK can send back-to-back frames continuously.

To read and write in the FPGA control registers, you must enter **hexadecimal values**.
For both the read and write commands, data and address.

Address	Function
0x20	Write 1 at bit index 0 to start the FPGA emission; write 0 to stop. > W 20 1 : Start the FPGA emission > W 20 0 : Stop the FPGA emission
0x21	Progressive Words Counter (not Frame Index) <u>start value</u> , by default 0 The Word counter start value is applied when the streaming starts (W 20 1) Keep the default value (0)
0x22	Write Counter (32 bits words inside the payload) Maximum Value. Emission stops after reaching the max value which is the number of 32 bits words sent. Use 0 for endless emitting Default = 0 > W 22 0 : emission will be continuous (must be stopped manually). > W 22 80000000 : sends 2^{31} words = 16G Bytes ie approx 13 seconds at full speed
0x23	Value of <u>Supplemental</u> Inter-Frame Gap between two frames, ie extra pause time between frames, expressed in clock cycles (5 ns in this design). Default = $1e6 = 5 \text{ ms} = \sim 1.6 \text{ Mbytes/s}$ only (<u>extremely</u> slow) Use small values to increase emitting speed. Use 0 for maximum speed. > W 23 0 : minimum IFG = maximum speed, back-to-back frames.
0x24	Emitted frames Length value. Up to 8952. Use 1452 (dec) = 5AC (hex) for non-jumbo RMTU (for example) The Reference Design uses 8192 (Jumbo frames) > W 24 2000 : the FPGA will generate Jumbo frames with a payload of 8192 bytes
0x25	Pause between two Data Valid at the 10GEDEK Tx input. Default is 0. Not useful in this context (keep 0)

VI - Conclusion

This document explains how to use our 10GEDEK Reference Design on the Polarfire Evaluation kit.

The port of our IPs to this kit was a **globally positive experience** :

- Using a new **Transceiver** is always a bit challenging since these blocks have many functions, many connections, they are internally quite complex and they are highly configurable.
We have a lot of experience with a lot of different families and vendors, and this went quite well.
We were able to perform the HDL **simulation** including the transceiver quite easily using the simulator(s) coming with Libero, and this is indeed an important feature.
- The **tools** are doing the job, Synplify Pro is a good synthesizer, scripting works but requires a bit of caution to keep every piece up to date and some juggling with Tcl scripts especially for IPs shared between projects. But no complain here either.
- We had no **timing convergence** issue. Our RTL code is generic and optimized so we didn't really expect timing problems and we had none.
- All our work on the kit was achievable using the **free Libero License** (Silver), and this is a definite "plus". We feared that a commercial license would be needed to use the Microchip Core10GBaseR_PHY but it was not the case and the free license was fine. We would have used our own logic for this block, but it makes more sense using the Microchip IP which is free to use for Licensed users.
Note : the **free** license supports the **FPGA used on the kit**. Other Polarfire FPGAs will require a paid-mode license.
- On the less positive side, we found that the **external Microchip PLL** used on the kit was very powerful but very complex to program. Creating the *HDL block to program this PLL* as we needed with the proper (very large) list of commands was an unexpected challenge, but it worked.
Likewise, porting our 1G GEDEK on the **Microchip Gigabit Phy** (through sgmi & CDR) was difficult and we stopped with a design that didn't always start properly synchronized. We never had demand to support this Phy so we put this effort on hold (we will address the last marginal issue if/when needed).

We conclude that the Polarfire family is a valid platform to host our IPs or our designs.

For any type question about this document, please contact us by **E-mail** at info@alse-fr.com.



Advanced Logic Synthesis for Electronics
A.L.S.E. - <http://www.alse-fr.com>