E-link Interface firmware UG

FELIX project svn version 4400

1. Overview

This document describes an example E-link interface user module intended to communicate with FELIX. It incorporates a transmitter and receiver for 80Mbps (10b8b / HDLC encoding / no encoding), 160Mbps (10b8b encoding / no encoding) or 320Mbps (10b8b encoding / no encoding) e-links. The e-link speed is defined using generic parameters.

The general structure of the user example module is shown in Figure 1.

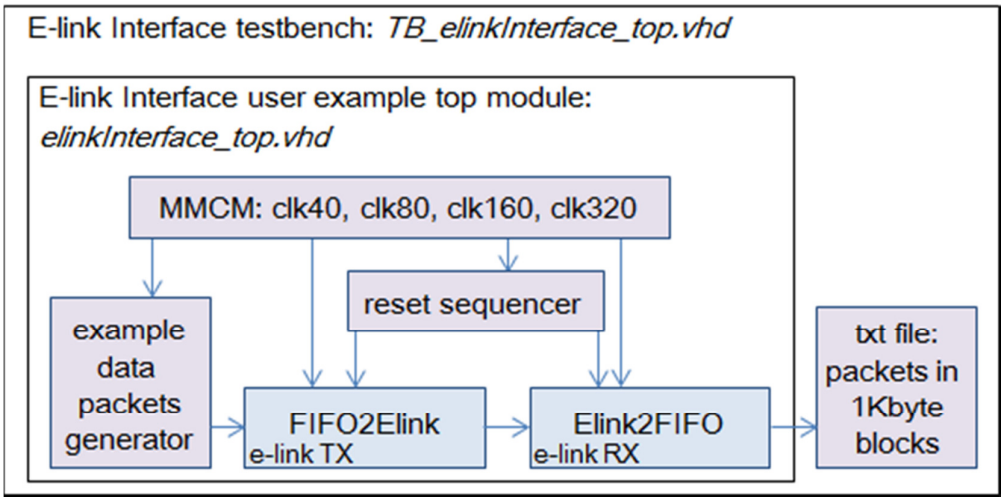

Figure 1: General structure

FIFO2Elink and Elink2FIFO modules can be used as an interface to a separate physical e-link or directly to a GBT frame. Configuration is defined by generics / constants in the package file: *elinkInterface_package.vhd*.

In *elinkInterface_top.vhd* e-link TX and e-link RX are connected using all available interfaces when the actual active interface is defined by generic parameters.

The simulation test bench, *TB_elinkInterface_top.vhd,* generates *the* output file, *elink_data_16bit.txt*, in the current simulation directory of the Vivado project. Note that the data is available at the output in complete 1 Kbyte blocks including block header and packet trailers (in FELIX' internal format). Consequently, the simulation run duration must be long enough to allow the accumulation of 1 Kbyte of data (in the example, more than 100 us).

To help guarantee consistency with the FELIX FPGA firmware, complete modules from FELIX are used, with unnecessary functions disabled, but not explicitly removed.

2. E-link transmitter: FIFO2Elink

2.1 FIFO2Elink interface options

FIFO2Elink module can interface serial e-link of 80, 160 or 320 Mbps or a GBT frame, 2, 4 or 8 bits at 40MHz.

For example, to assemble a single GBT frame, one can use several various rate e-links, see Figure 2.

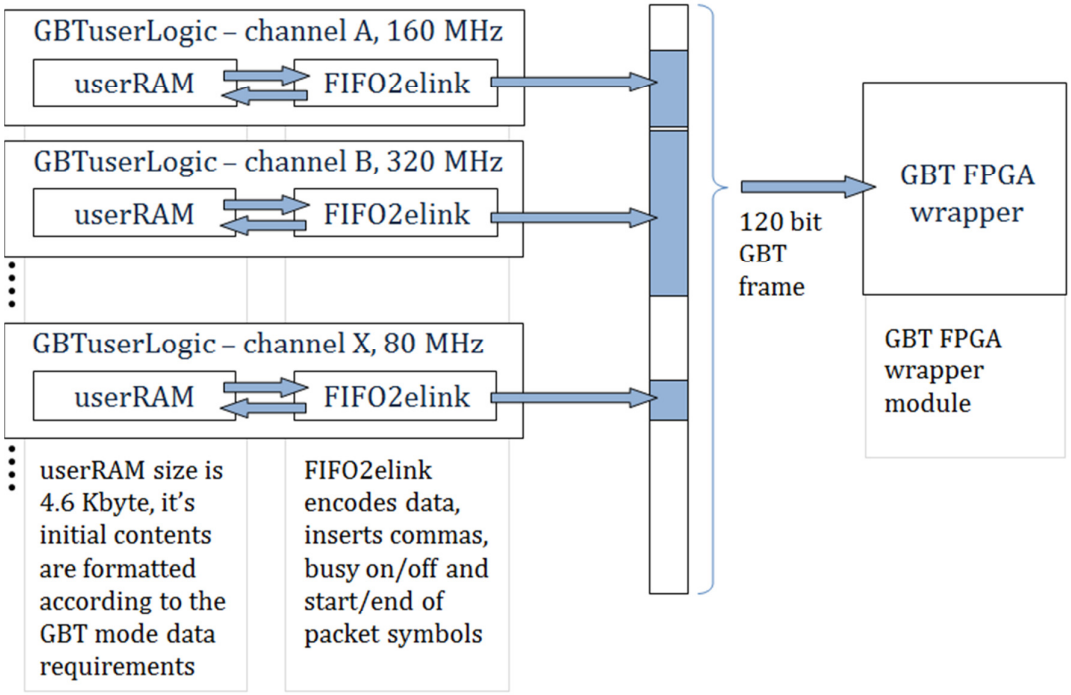

Figure 2: GBT frame assembly example

2.2 FIFO2Elink module VHDL entity

FIFO2Elink functions include:

- Encoding and transmitting user data according to generic settings
- Transmitting the packet boundary symbols (sop/eop)
- Transmitting the idle/coma when the user FIFO is empty

2.3 FIFO2Elink entity description

```
entity FIFO2Elink is
generic (
   OutputDataRate  : integer := 80;   -- 80/160/320, e-link speed in Mbps
   elinkEncoding      : std_logic_vector (1 downto 0)   -- "00"-direct data
   );                                                   -- "01"-8b10b encoding
                                                        -- "10"-HDLC encoding (80Mbps only)
port (
   clk40       : in  std_logic;
   clk80       : in  std_logic;                         -- phase aligned clocks
   clk160      : in  std_logic;
   clk320      : in  std_logic;
   rst         : in  std_logic;                         -- rst has to be 'high' when fifo_flush is 'high'
   fifo_flush  : in  std_logic;                         -- rst/fifo_flush sequence has to be issued at the startup
   ------
   efifoDin    : in   std_logic_vector (17 downto 0);   -- [data_code,2bit][data,16bit]
   efifoWe     : in   std_logic;                        -- standard FIFO interface
   efifoPfull  : out std_logic;
   efifoWclk   : in   std_logic;
   ------
   --
   --
   --
   --
   DATA1bitOUT : out std_logic;   -- to e-link, serialized output
   --
   elink2bit      : out std_logic_vector (1 downto 0);   -- 2 bits @ clk40, can interface GBT frame
   elink4bit      : out std_logic_vector (3 downto 0);   -- 4 bits @ clk40, can interface GBT frame
   elink8bit      : out std_logic_vector (7 downto 0)    -- 8 bits @ clk40, can interface GBT frame
   ------
   );
end FIFO2Elink;
```

data_code,2bit	data,16bit
"00"-data	data
"01"-end of packet	not used
"10"-start of packet	not used
"11" -comma (no use)	not used

3 E-link receiver: Elink2FIFO

3.1 Elink2FIFO interface options

Elink2FIFO module can interface serial e-link of 80, 160 or 320 Mbps or a GBT frame, 2, 4, 8 or 16 bits at 40MHz.

For example, to interface a single GBT frame, one can use several various rate e-links, see Figure 2.

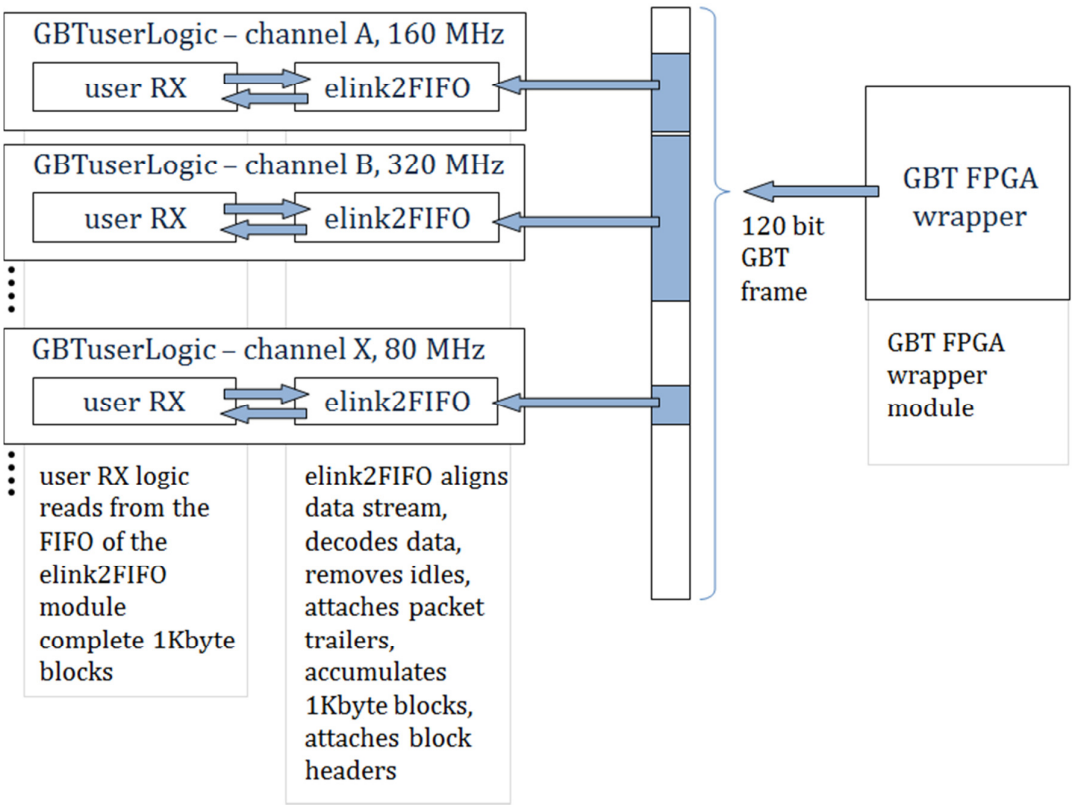

Figure 3: GBT frame interface example

3.2 Elink2FIFO module VHDL entity

Elink2FIFO functions include:

- Aligning the data stream according to encoding (see Appendix)
- Decoding the received data according to generic settings
- Recognizing the packet boundary symbols (sop/eop)
- Accumulating the received data in 1Kbyte blocks attaching block header and packet trailers

3.3 Elink2FIFO VHDL entity description

```
entity Elink2FIFO is
generic (
   InputDataRate : integer := 80;   -- 80/160/320/640, e-link speed in Mbps
   elinkEncoding   : std_logic_vector (1 downto 0)  -- "00"-direct data
   );                                               -- "01"-8b10b encoding
                                                    -- "10"-HDLC encoding (80Mbps only)
port (
   clk40       : in  std_logic;
   clk80       : in  std_logic;
   clk160      : in  std_logic;                     -- phase aligned clocks
   clk320      : in  std_logic;
   rst         : in  std_logic;
   fifo_flush  : in  std_logic;                     -- rst has to be 'high' when fifo_flush is 'high'
   ------                                           -- rst/fifo_flush sequence has to be issued at the startup
   DATA1bitIN  : in  std_logic;  -- from e-link, serial input @ 80, 160 or 320 Mbps only
   elink2bit     : in  std_logic_vector (1 downto 0) := (others=>'0');  -- 2 bits of GBT frame @clk40
   elink4bit     : in  std_logic_vector (3 downto 0) := (others=>'0');  -- 4 bits of GBT frame @clk40
   elink8bit     : in  std_logic_vector (7 downto 0) := (others=>'0');  -- 8 bits of GBT frame @clk40
   elink16bit    : in  std_logic_vector (15 downto 0) := (others=>'0'); -- 16 bits of GBT frame
   ------
   efifoRclk    : in    std_logic;
   efifoRe      : in    std_logic;                  -- FIFO interface *
   efifoHF      : out std_logic;
   efifoDout    : out std_logic_vector (15 downto 0)
   ------
   );
end Elink2FIFO;
```

* FIFO interface (see Table 1.)

1. The data is available at the output in complete 1Kbyte blocks including block header and packet trailers.
2. When **efifoHF** output becomes 'high', the read of 1Kbyte block should be initiated (512 read cycles).
3. **efifoRclk** has to be 160MHz or higher.

Table 1. FELIX 1 Kbyte block internal format

Value	**Description**
0x0000	Block header (15 downto 0)
0xABCD	Block header (31 downto 16)
16-bit word	packet0.word0, 16 bit
16-bit word	packet0.word1, 16 bit
…	…
16-bit word	packet0.wordN, 16 bit
[trailer(15:13),type] [trailer(12),truncation] [trailer(11),error] [trailer(10),reserved] [trailer(9:0),sub-packet length]	packet0.trailer, 16 bit when trailer type is: "000": null trailer "001": first sub-packet "010": last sub-packet "011": whole sub-packet "100": middle sub-packet "101": e-link timeout trailer "110": reserved "111": out-of-band message
16-bit word	packet1.word0, 16 bit
…	…
16-bit word	packet1.wordM, 16 bit
[011][0][0][0][2*(M+1),10bit]	packet1.trailer, 16 bit
16-bit word	packet1.word0, 16 bit
…	…
[001][0][0][0][2*(i+1),10bit] if (i+1) words of packet**x** were written here but the rest continues in the next 1Kbyte block	packet**x**.trailer, 16 bit

4 Appendix

Table 2. FELIX 8b/10b encoding symbols

symbol	binary string	description
K28.5	0011111010 / 1100000101	comma / idle
K28.1	0011111001 / 1100000110	start of packet
K28.6	0011110110 / 1100001001	end of packet
K28.2	0011110101 / 1100001010	start of busy
K28.3	0011110011 / 1100001100	end of busy

Table 3. FELIX HDLC encoding symbols

symbol	binary string	description
0xFF	11111111	error / idle
0x7E	01111110	start / end of packet