Open Source Event Receiver

Open Hardware Workshop EPICS Meeting

Argonne, June 2018

Jukka Pietarinen

Micro-Research Finland Oy

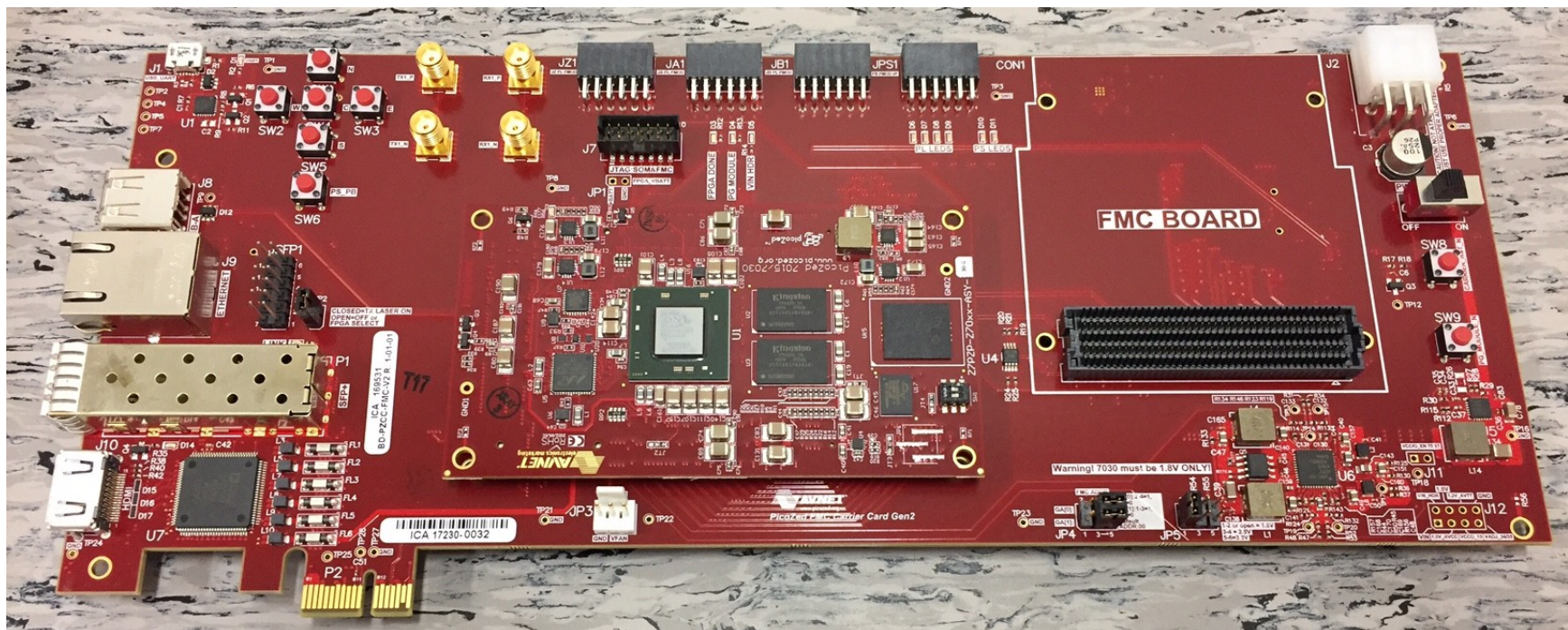# Open Source Event Receiver - Introduction

- What is the Open Source Event Receiver

  - Basic building block required to build devices receiving the MRF Timing protocol including but not limited to Delay Compensation capability

- What is it not

  - it is not a replacement firmware for current MRF products

  - it is not a complete Event Receiver with MRF product compatible register map

    - No bus/register interface

    - No pulse generators

    - etc.

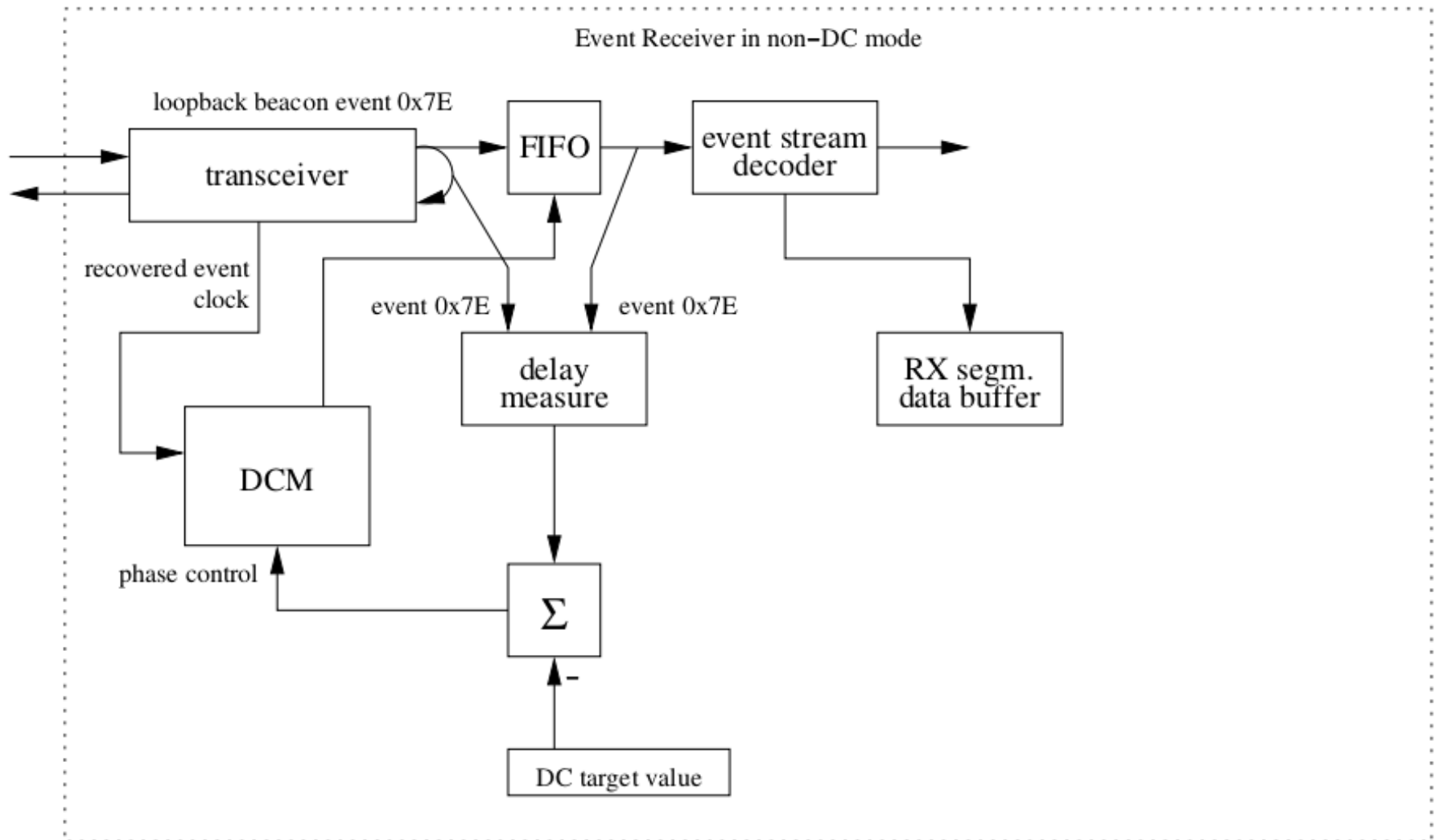# Open Source Event Receiver - Requirements

- Hardware
  - Xilinx Kintex-7 based FPGA with GTX transceivers **ONLY!**
  - Zynq 7Z030 is Kintex-7 based
  - SFP Transceiver
  - Reference clock for GTX
  - Example design built for
    - Avnet PicoZed 7Z030
    - Avnet PicoZed FMC Carrier Card V2
- Software
  - Xilinx Vivado 2017.4 (Free WebPack version is sufficient)
- Xilinx programming cable
  - e.g. Platform Cable USB II

Avnet PicoZed FMC carrier with 7Z030 SOM

- Avnet PicoZed AES-Z7PZ-7Z030-SOM-G
- Avnet PicoZed FMC carrier AES-PZCC-FMC-V2-G
  - Zynq 7Z030 incorporates
    - Kintex-based FPGA core
    - Four GTX transceivers
    - Dual-core ARM Cortex-A9
- This kit has everything from the hardware point of view to be used as an event receiver

Open Source Event Receiver - non-DC mode

Event Receiver in non-DC mode

loopback beacon event 0x7E

transceiver

FIFO

event stream decoder

recovered event clock

event 0x7E

event 0x7E

delay measure

RX segm. data buffer

DCM

phase control

Σ

−

DC target value

Reference Design Structure

zynq_top.vhd - Design top level

    evr_dc.vhd - Event Receiver top level

        transceiver_dc_k7.vhd - GTX Transceiver instantiation

        delay_measure.vhd - Delay measurement

           average.vhd
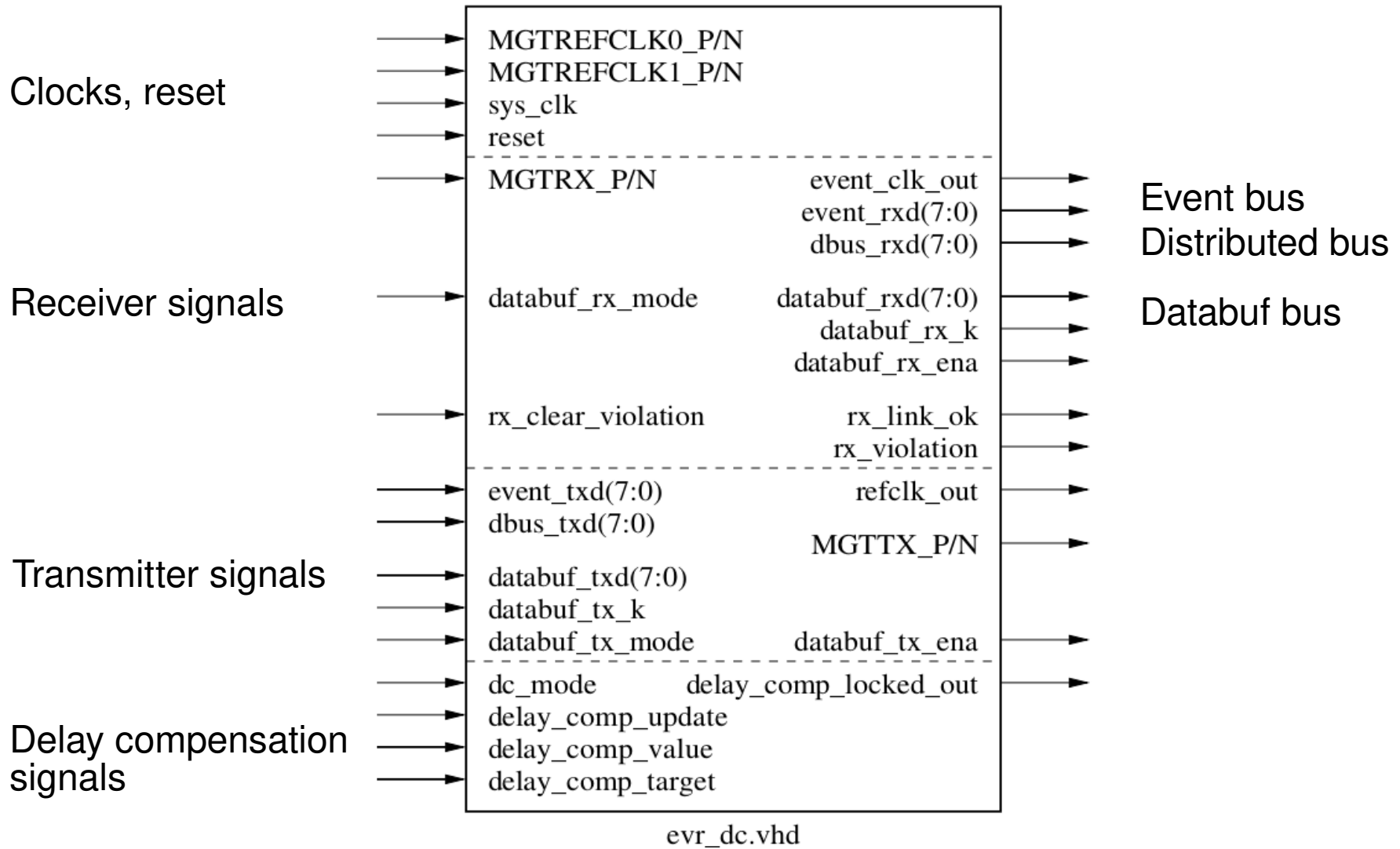
        delay_adjust.vhd - DCM control

    databuf_rx_dc.vhd - Segmented data buffer receiver

        evr_pkg.vhd

        buf_bsram.vhd

zynq.xdc - Constraints file

Open Source Event Receiver

Clocks, reset

MGTREFCLK0_P/N
MGTREFCLK1_P/N
sys_clk
reset

MGTRX_P/N

Receiver signals

databuf_rx_mode

rx_clear_violation

event_clk_out
event_rxd(7:0)
dbus_rxd(7:0)

databuf_rxd(7:0)
databuf_rx_k
databuf_rx_ena

rx_link_ok
rx_violation

Event bus
Distributed bus

Databuf bus

Transmitter signals

event_txd(7:0)
dbus_txd(7:0)

databuf_txd(7:0)
databuf_tx_k
databuf_tx_mode

refclk_out

MGTTX_P/N

databuf_tx_ena

Delay compensation signals

dc_mode
delay_comp_update
delay_comp_value
delay_comp_target

delay_comp_locked_out

evr_dc.vhd

Minimum Configuration

- Connect reference clock to one reference clock input MGTREFCLKx_P/N

- Connect system clock to sys_clk

- Connect transceiver receive signals to MGTRX_P/N

- Tie databuf_rx_mode and databuf_tx_mode high '1'

- Tie dc_mode low '0' (non-DC mode)

- Set delay_comp_target to a fixed value > 0x00050000, this value can be used to fine tune the "group delay" of the whole EVR. In non-DC mode this value sets depth of the EVR input FIFO.

- Tie other (unused) inputs low '0'

Receiving Events

- Event codes are presented on `event_rxd` in `event_clk` clock domain

- When `event_rxd` is not 0x00 there is an active event code

- `link_ok` can be used to check link status

```
if rising_edge(event_clk) then
    if event_rxd = X"01" then
        -- event code X"01" received
    end if;
end if;
```
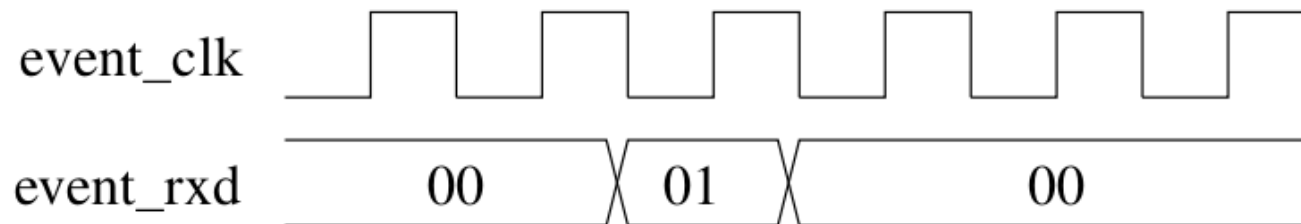
Receiving Distributed Bus Bits

- Distributed bus bits presented on `dbus_rxd` in `event_clk` clock domain

- When databuf mode is enabled `dbus_rxd` is updated every other clock cycle, when databuf mode is disabled `dbus_rxd` is updated every clock cycle

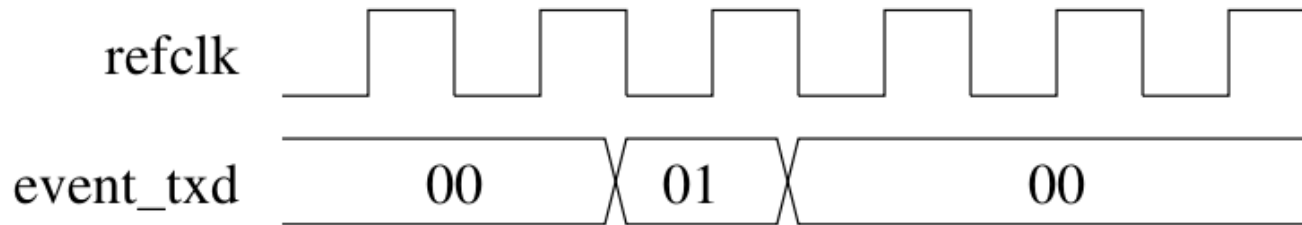- `link_ok` can be used to check link status

Databuf bus

- Databuf bus is used to transfer data packets on `databuf_rxd`, `databuf_rx_k, databuf_ena` in `event_clk` clock domain
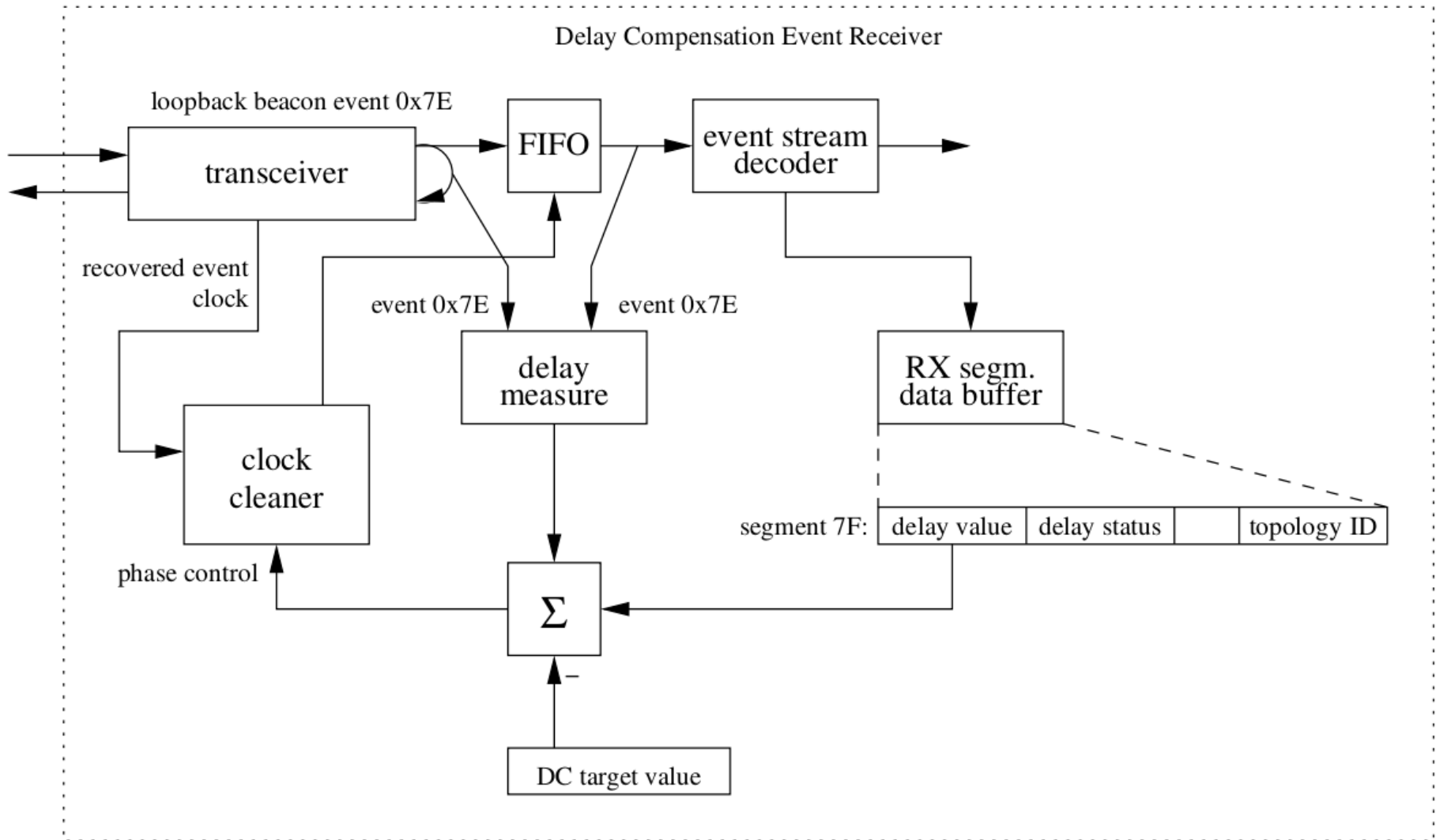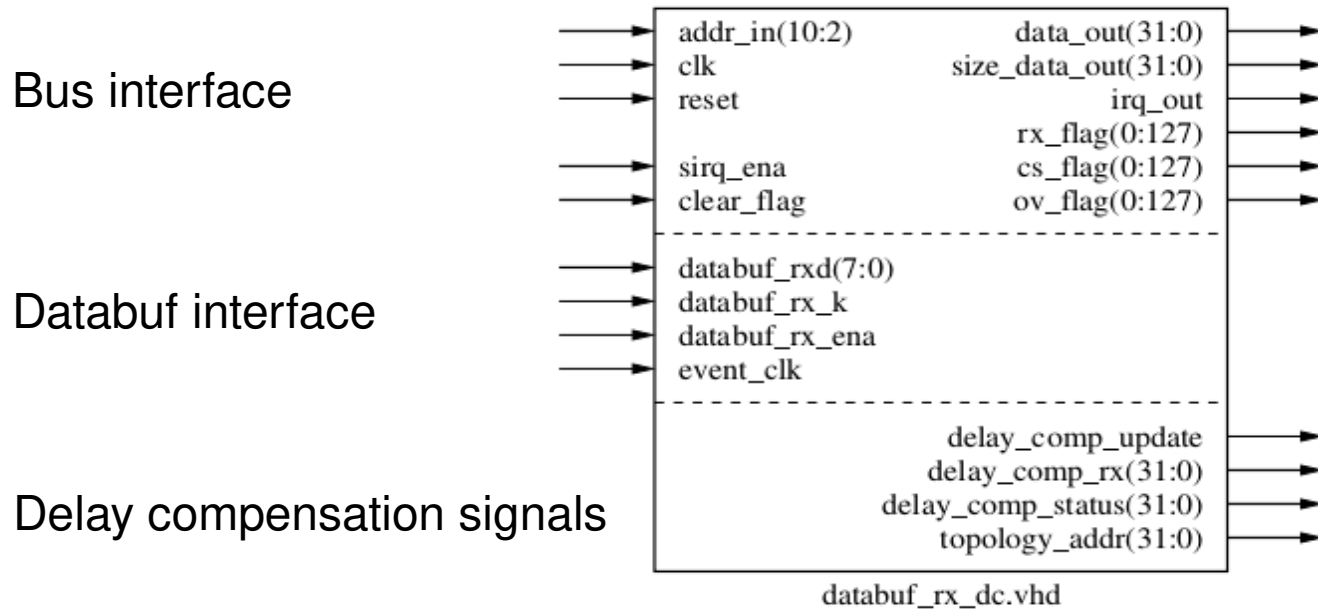
- Databuf mode has to be enabled
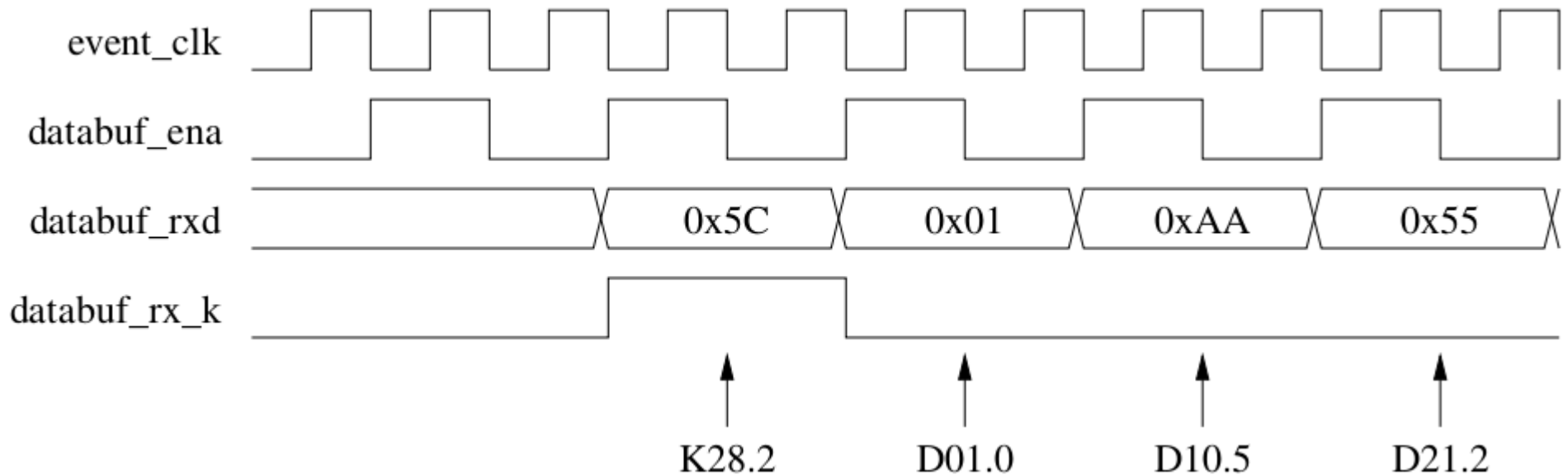
Sending Events

- Event codes are presented on `event_txd` in `refclk` clock domain

- When `event_txd` is not 0x00 the event code is sent out on the rising edge of `refclk`

Open Source Event Receiver - DC mode

Delay Compensation Event Receiver

loopback beacon event 0x7E

transceiver

FIFO

event stream decoder

recovered event clock

event 0x7E

event 0x7E

clock cleaner

delay measure

RX segm. data buffer

phase control

segment 7F: | delay value | delay status | | topology ID |

Σ

−

DC target value

Adding delay compensation

Bus interface

Databuf interface

Delay compensation signals

```
addr_in(10:2)                    data_out(31:0)
clk                          size_data_out(31:0)
reset                                    irq_out
                                 rx_flag(0:127)
sirq_ena                         cs_flag(0:127)
clear_flag                       ov_flag(0:127)
---------------------------------------------------
databuf_rxd(7:0)
databuf_rx_k
databuf_rx_ena
event_clk
---------------------------------------------------
                             delay_comp_update
                             delay_comp_rx(31:0)
                          delay_comp_status(31:0)
                             topology_addr(31:0)
```

databuf_rx_dc.vhd

- `delay_comp_update` has to be connected to `delay_comp_update` of the evr_dc block

- `delay_comp_rx` has to be connected to `delay_comp_value` of the evr_dc block

Building Reference Design with Vivado

- Getting Sources
  - git clone https://github.com/jpietari/mrf-openevr
- Building Vivado project
  - cd mrf-openevr
  - vivado -mode tcl
    - Vivado% source ./openevr.tcl
    - Vivado% quit
- Synthesis/implementation/creating bitstream
  - Launch vivado in GUI mode
    - Vivado
      - Open project openevr/openevr.xpr
      - Generate bitstream
- FPGA configuration
  - Hardware manager
 - connect to target
  - download

Reference Design Features

- LEDs
  - LED1 (D6), rx_violation
  - LED2 (D7), link_ok
  - LED3 (D8), flashed quickly on received 0x01 event code
  - LED4 (D9), flashes slowly
- Pushbuttons
  - SW1 (N), rx_clear_violation
  - SW2 (S), tx_reset
  - SW3 (E), sys_reset
- The event link can be looped back to itself (connect fiber patch cable from SFP TX to SFP RX)
  - however due to GTX internals and running the transmitter and receiver from the same clock source you will need to press tx_reset several times before link gets established, this applies only to self-loopback. Theoretically, chances of establishing link is $1/20^{th}$ resets.
- Sending out event code 0x01 at a fixed rate, few Hz, received event shows on LED3 (D8)
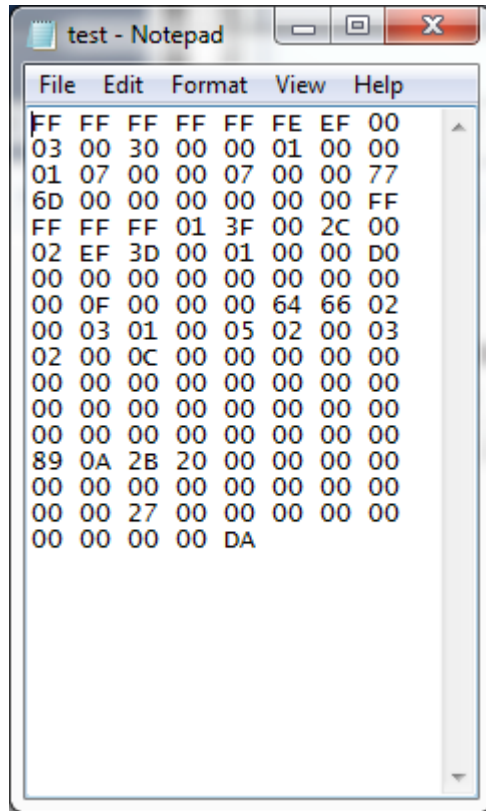- Two integrated logic analyzer (ILA) cores instantiated

Avnet PicoZed FMC Carrier V2 Board Reference Clock

- Avnet PicoZed FMC Carrier V2 has a programmable IDT 8T49N242 clock synthesizer.

  - Configuration resides on an EEPROM

  - Reference design available from Avnet to program EEPROM

- Software Tools

  - IDT Timing Commander

  - IDT 8T49N24x Timing Commander Personality File

  - Xilinx Vivado and SDK 2015.4 (exact version required)

- Getting Sources

  - http://picozed.org/support/design/13076/106

  - Transceiver Clock Programming Reference Design

IDT Timing Commander

IDT Timing Commander

test - Notepad
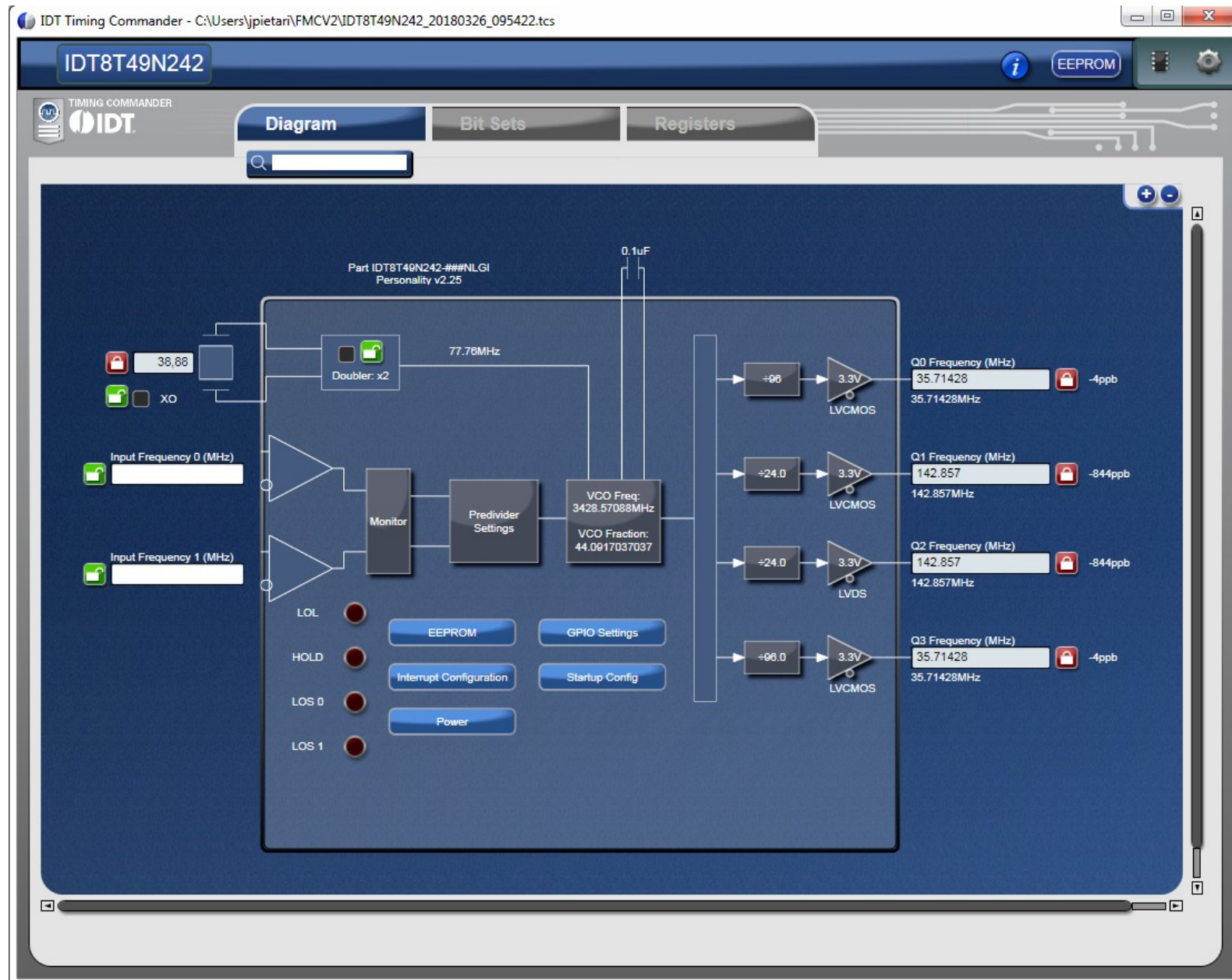
| File | Edit | Format | View | Help |

```
FF FF FF FF FF FE EF 00
03 00 30 00 00 01 00 00
01 07 00 00 07 00 00 77
6D 00 00 00 00 00 00 FF
FF FF FF 01 3F 00 2C 00
02 EF 3D 00 01 00 00 D0
00 00 00 00 00 00 00 00
00 0F 00 00 00 64 66 02
00 03 01 00 05 02 00 03
02 00 0C 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
89 0A 2B 20 00 00 00 00
00 00 00 00 00 00 00 00
00 00 27 00 00 00 00 00
00 00 00 00 DA
```

Copy table as a new configuration to file
C:\Avnet\hdl\Projects\pz_fmc2_valtest\software\pzcc_iic_eeprom_test\src\iic_eeprom_demo.c
Follow instruction in PizoZed_FMC2_Carrier_IDT_Clock_Programming_RefDes_2015_4.pdf

Evaluation Board Reference Clock

- Launch Vivado 2015.4
  - Open Project C:/Avnet/hdl/Projects/pz_fmc2_valtest/PZ7030_FMC2
  - Generate Bitstream
- Launch SDK 2015.4
  - Select Workspace C:\Avnet\hdl\Projects\pz_fmc2_valtest\PZ030_FMC2\pz_fmc2_valtest.sdk
  - Xilinx Tools -> Program FPGA, Click Program
  - Right-click pzcc_iic_eeprom_test
    - Run as -> Run Configurations
  - Click on Application tab
  - Search: pzcc_iic_eeprom_test.elf
  - Apply, Run

Further developments

- Adding more "bits and pieces" as configurable "plugins"

  - timestamping?

  - pulse generators?

  - bus interface for CPU access?

- Main idea is still to keep it as simple as possible

- VHDL is not a very convenient language for configurable designs - you have to use generate statements - no #ifdefs