Open Source Event Receiver

Timing Workshop EPICS Meeting Fall 2018

Australian Synchrotron, November 2018

Jukka Pietarinen

Micro-Research Finland Oy

Open Source Event Receiver - Introduction

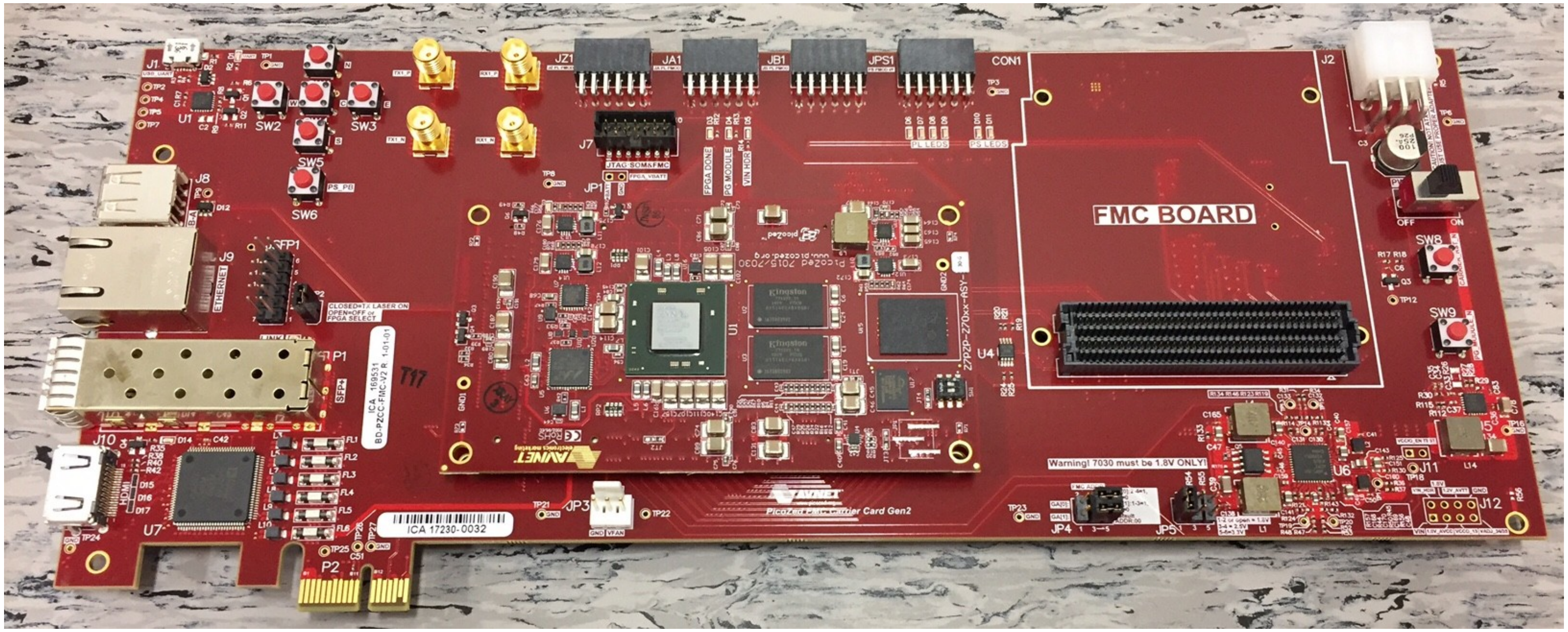
- What is the Open Source Event Receiver
  - Basic building block required to build devices receiving the MRF Timing protocol including but not limited to Delay Compensation capability
- What is it not
  - it is not a replacement firmware for current MRF products
  - it is not a complete Event Receiver with MRF product compatible register map
    - No bus/register interface
    - No pulse generators
    - etc.

Open Source Event Receiver - Requirements

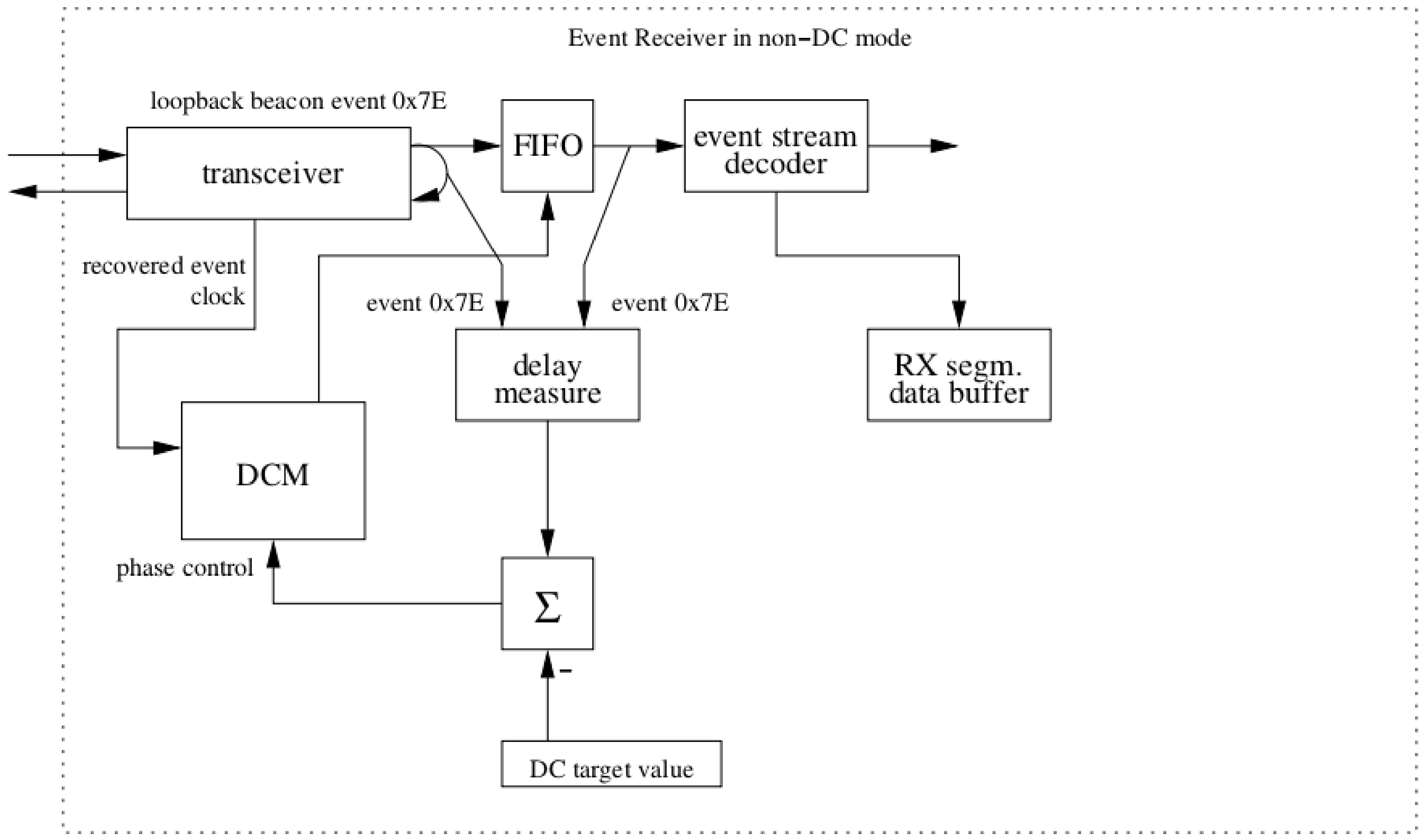
- Hardware
  - Xilinx Kintex-7 based FPGA with GTX transceivers **ONLY!**
  - Zynq 7Z030 is Kintex-7 based
  - SFP Transceiver
  - Reference clock for GTX
  - Example design built for
    - Avnet PicoZed 7Z030
    - Avnet PicoZed FMC Carrier Card V2
- Software
  - Xilinx Vivado 2017.4 (Free WebPack version 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



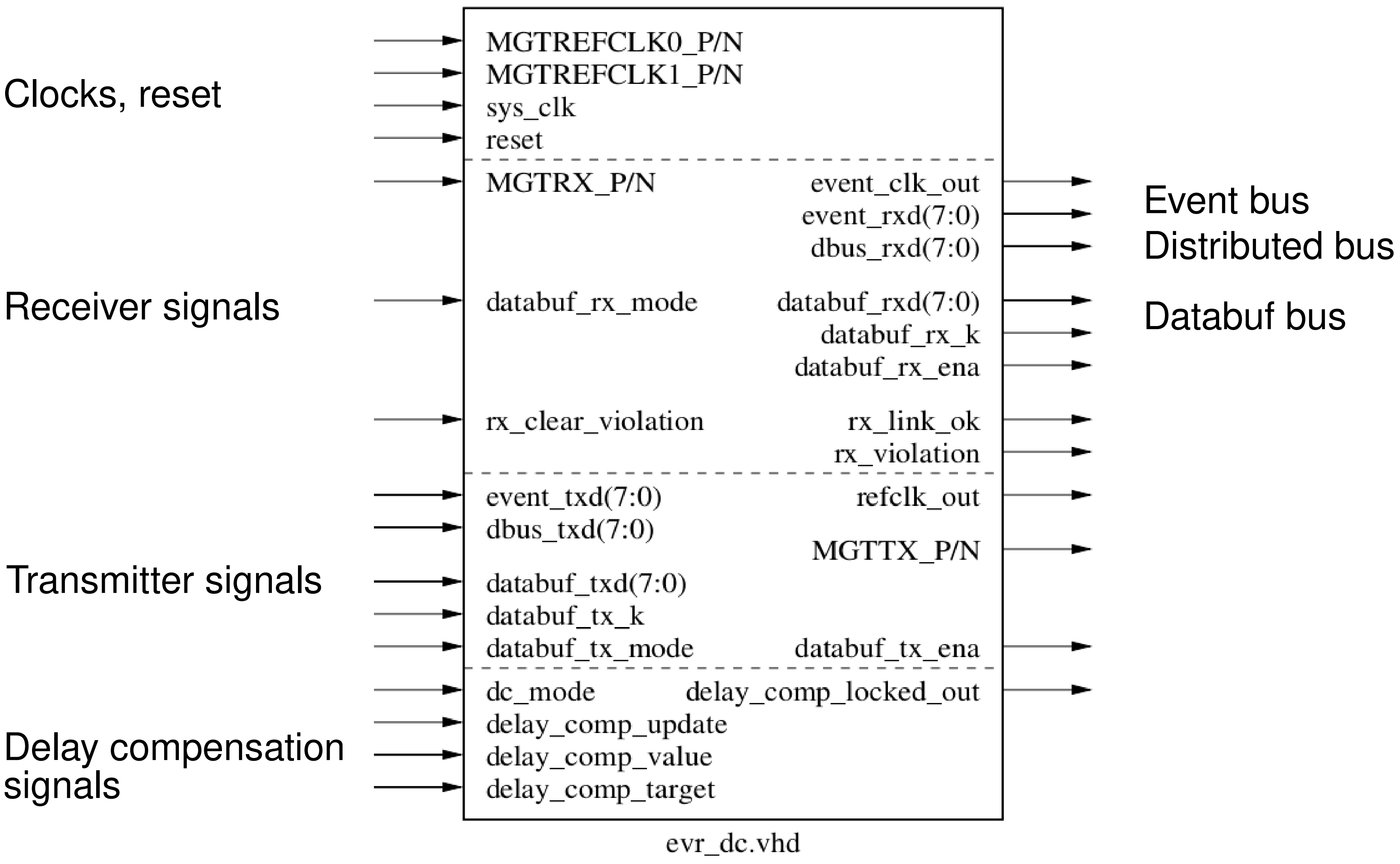
Open Source Event Receiver - non-DC mode



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



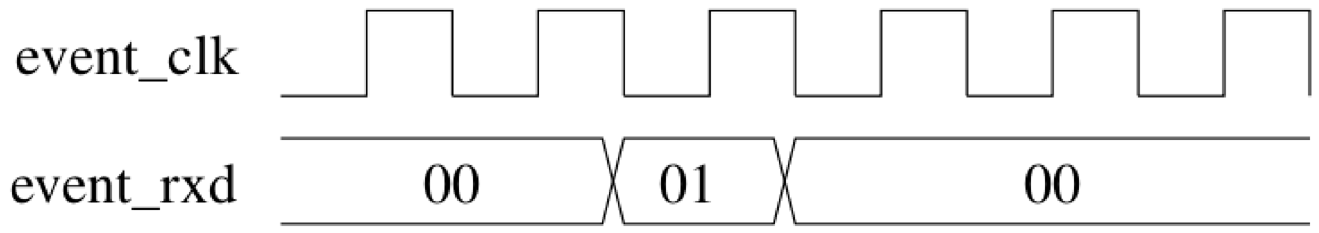
Minimum Configuration

- 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;
```

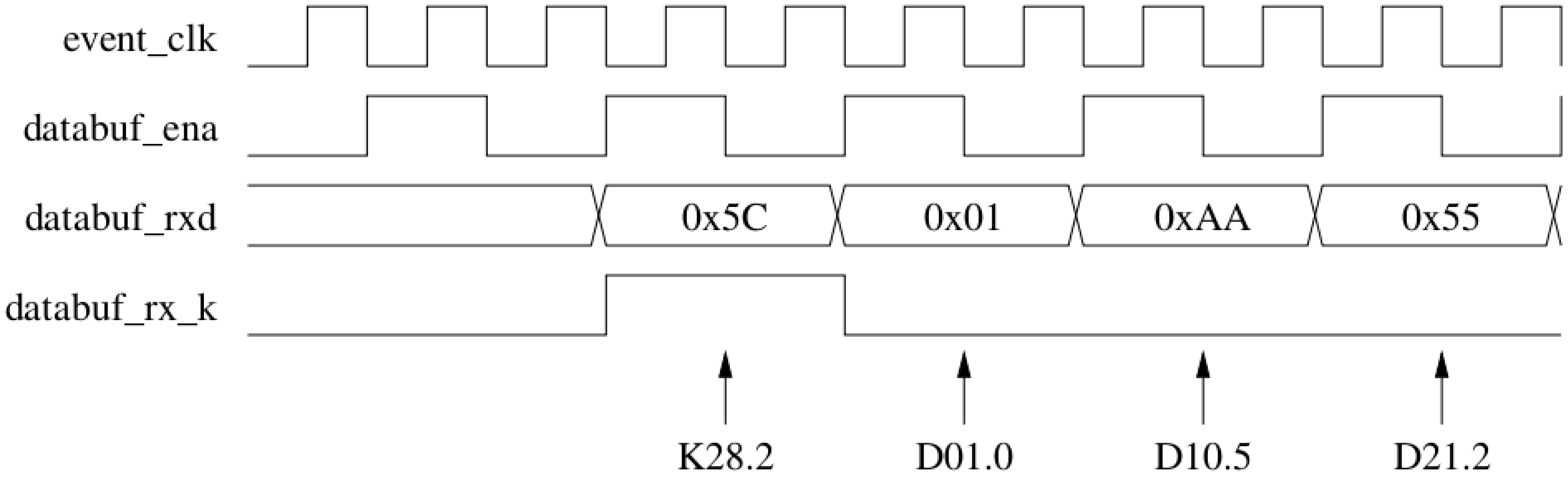


Receiving Distributed Bus Bits

- 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

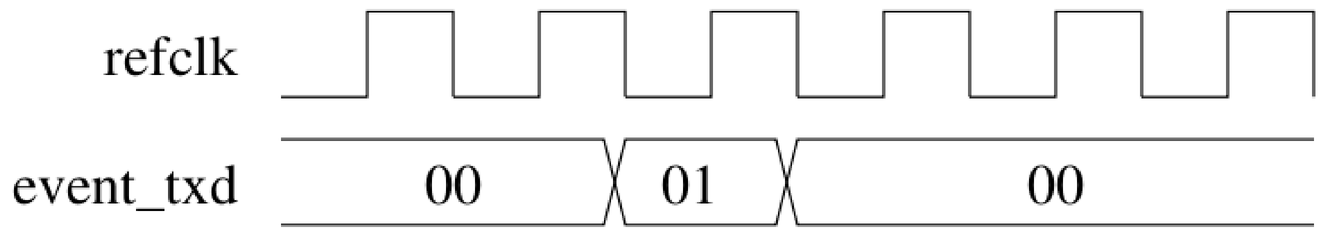
Databuf bus

- 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

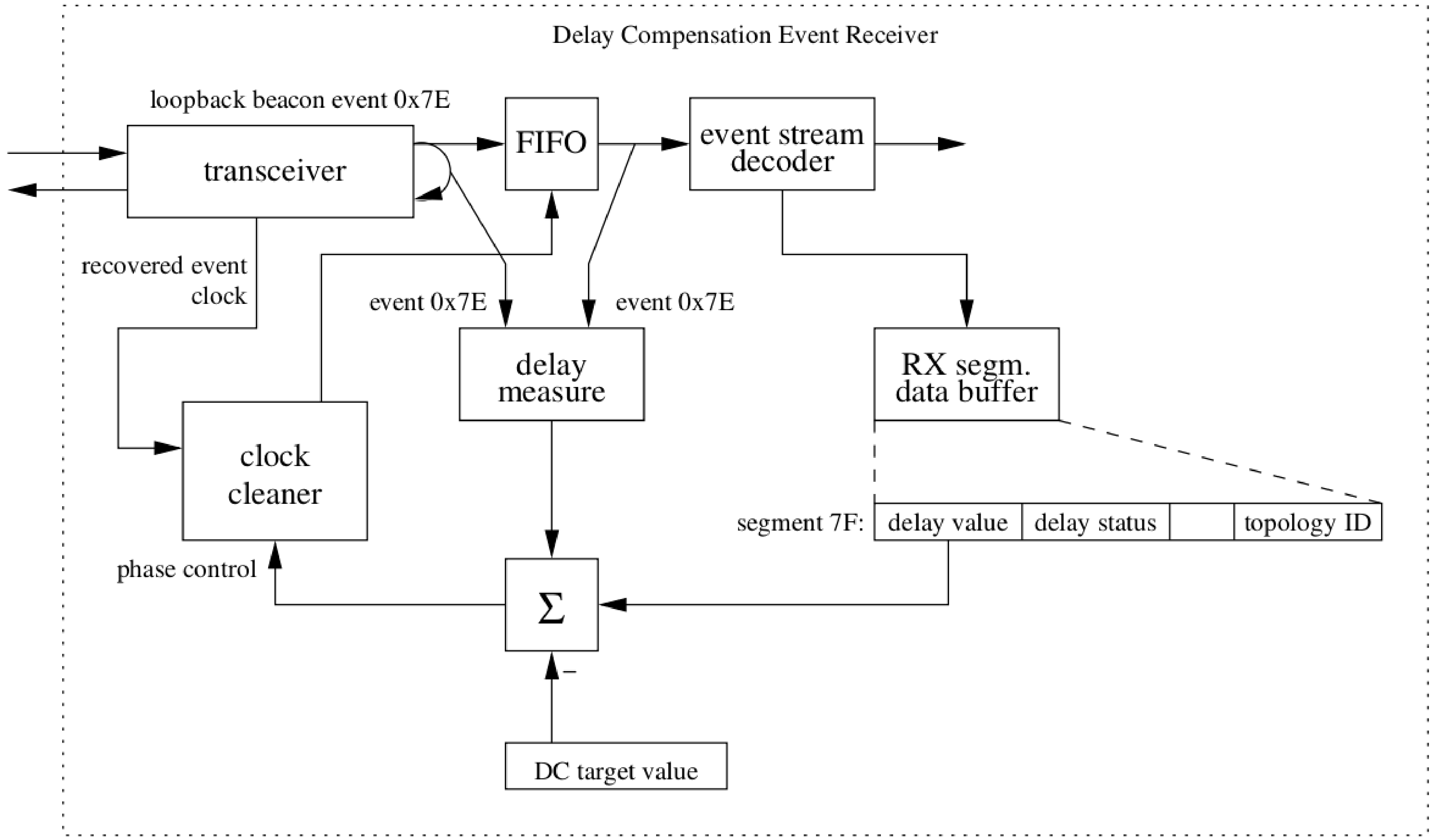


Sending Events

- 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

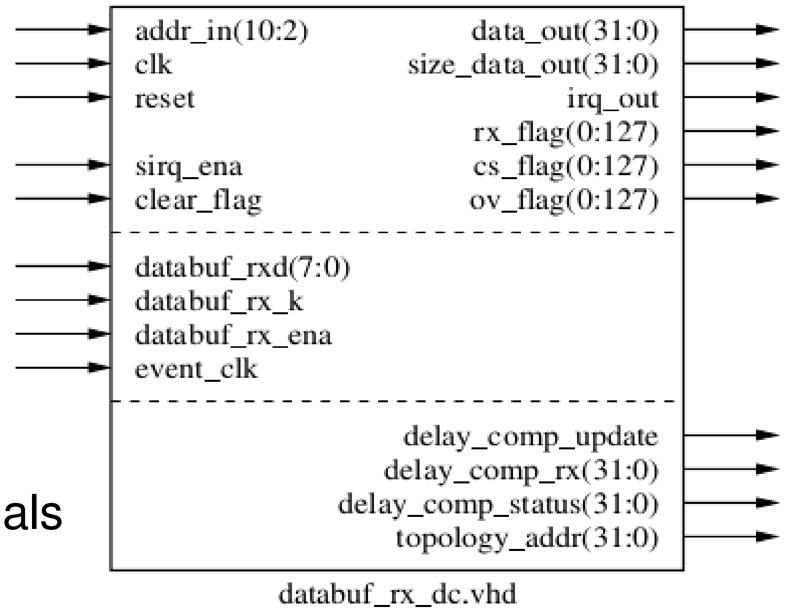


Adding delay compensation

Bus interface

Databuf interface

Delay compensation signals



- `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

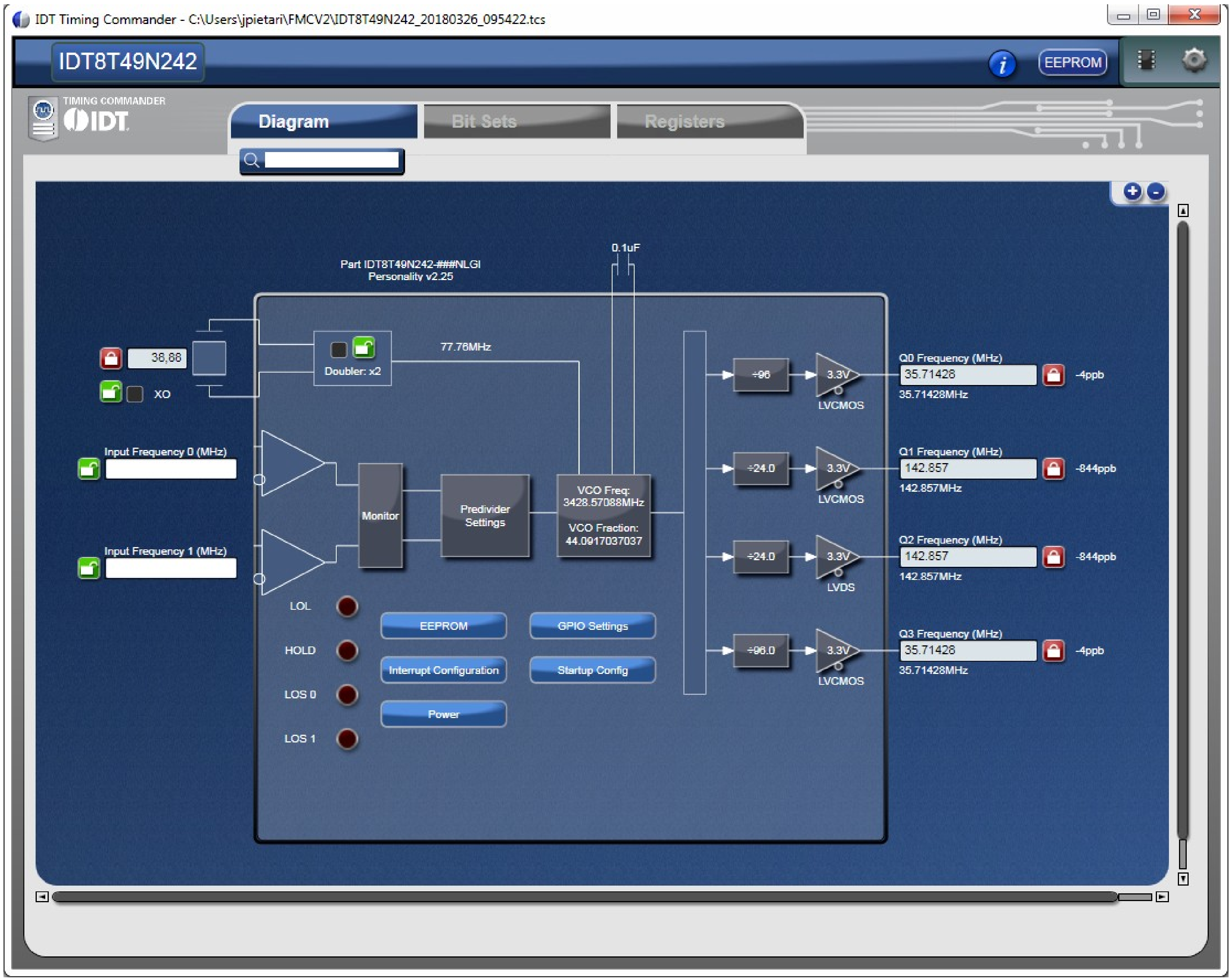
Reference Design Features

- 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 1/20th 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 - C:\Users\jpietari\FMCV2\IDT8T49N242_20180326_095422.tcs

IDT8T49N242

Part IDT8T49N242-###NLGI
Personality v2.25

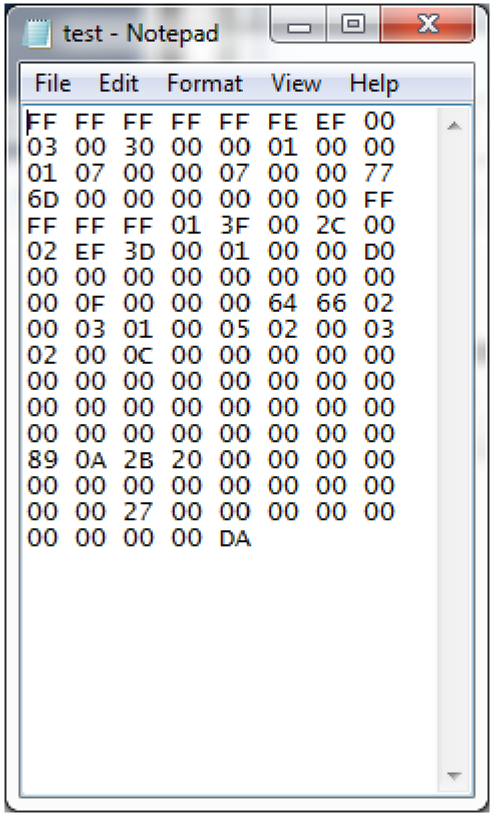
XO: 38.88

Doubler: x2 — 77.76MHz

VCO Freq: 3428.57088MHz
VCO Fraction: 44.0917037037

Output	Divider	Type	Frequency (MHz)	Actual	Error
Q0	÷96	3.3V LVCMOS	35.71428	35.71428MHz	-4ppb
Q1	÷24.0	3.3V LVCMOS	142.857	142.857MHz	-844ppb
Q2	÷24.0	3.3V LVDS	142.857	142.857MHz	-844ppb
Q3	÷96.0	3.3V LVCMOS	35.71428	35.71428MHz	-4ppb

IDT Timing Commander



```
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

Further developments (2)

- Meeting with PSI in September
- Requests:
  - Isolate the FPGA dependent part (MGT, Delay compensation adjustment) from the EVR core to allow easier porting to different platforms
  - Standardize interface between MGT and EVR core
- Modular approach
  - To allow easy integration of custom logic