

# Open Source Event Receiver

Timing Workshop EPICS Meeting Spring 2019

Jukka Pietarinen

Micro-Research Finland Oy

# Open Source Event Receiver - Introduction

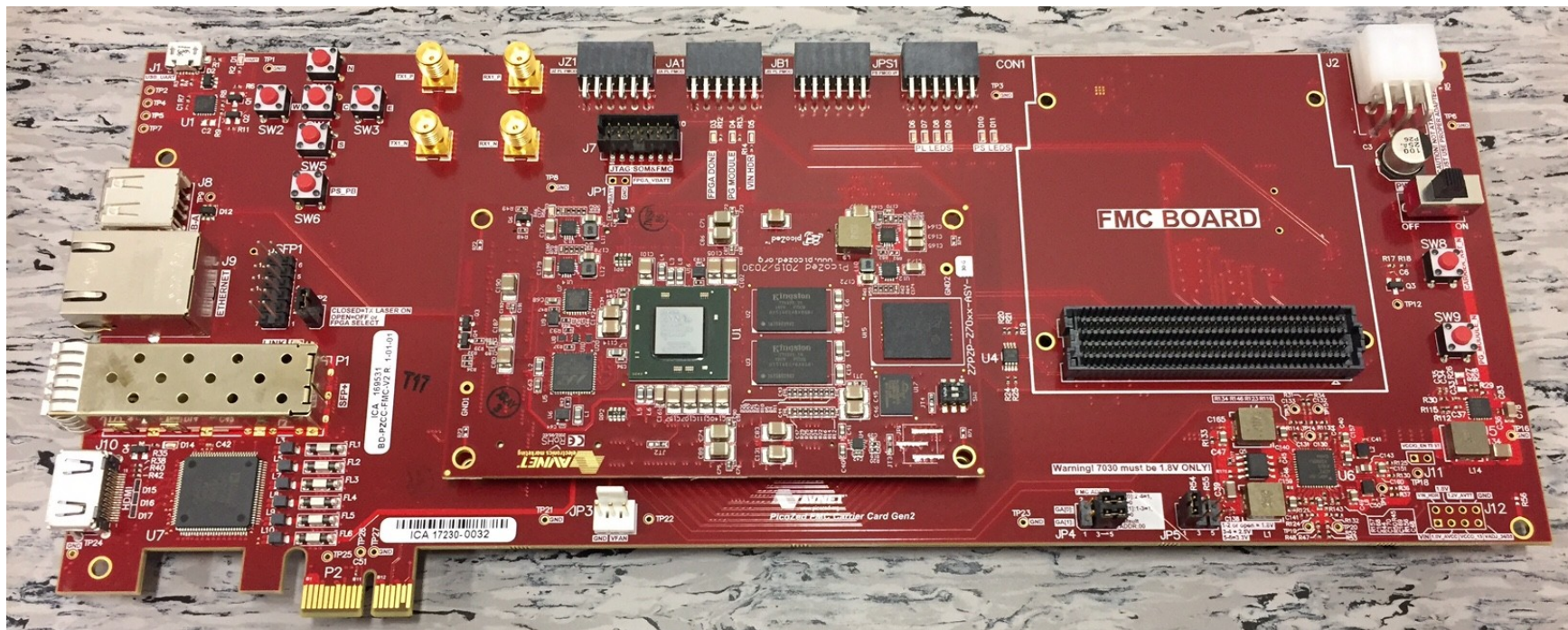
- What is the Open Source Event Receiver
  - Basic building block required to build devices receiving the MRF Timing protocol including but not limited to Delay Compensation capability
- What is it not
  - it is not a replacement firmware for current MRF products
  - it is not a complete Event Receiver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

# Open Source Event Receiver - Requirements

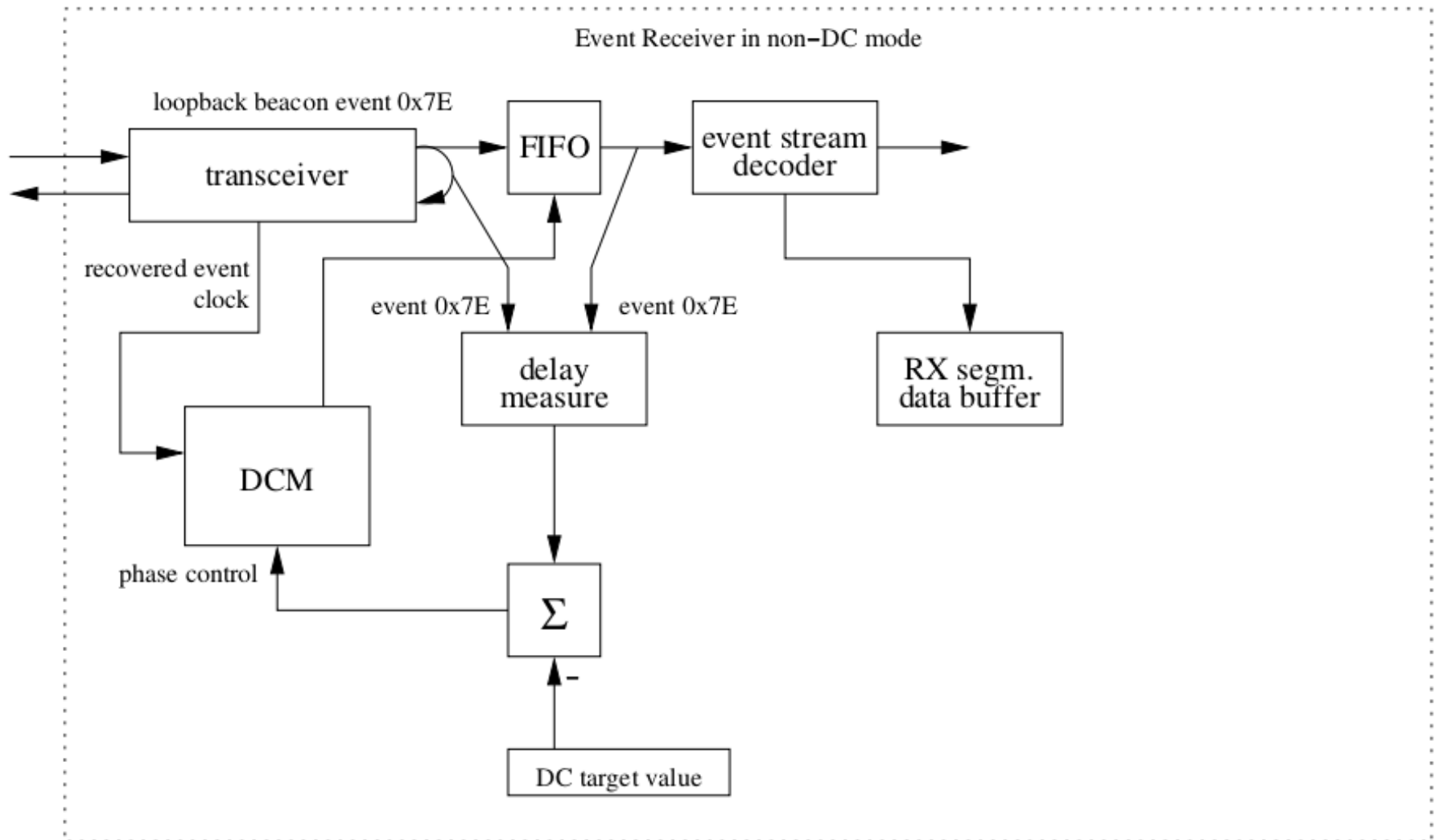
- Hardware
  - Xilinx Kintex-7 based FPGA with GTX transceivers **ONLY!**
  - Zynq 7Z030 is Kintex-7 based
  - SFP Transceiver
  - Reference clock for GTX
  - Example design built for
    - Avnet PicoZed 7Z030
    - Avnet PicoZed FMC Carrier Card V2
- Software
  - Xilinx Vivado 2017.4 (Free WebPack version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

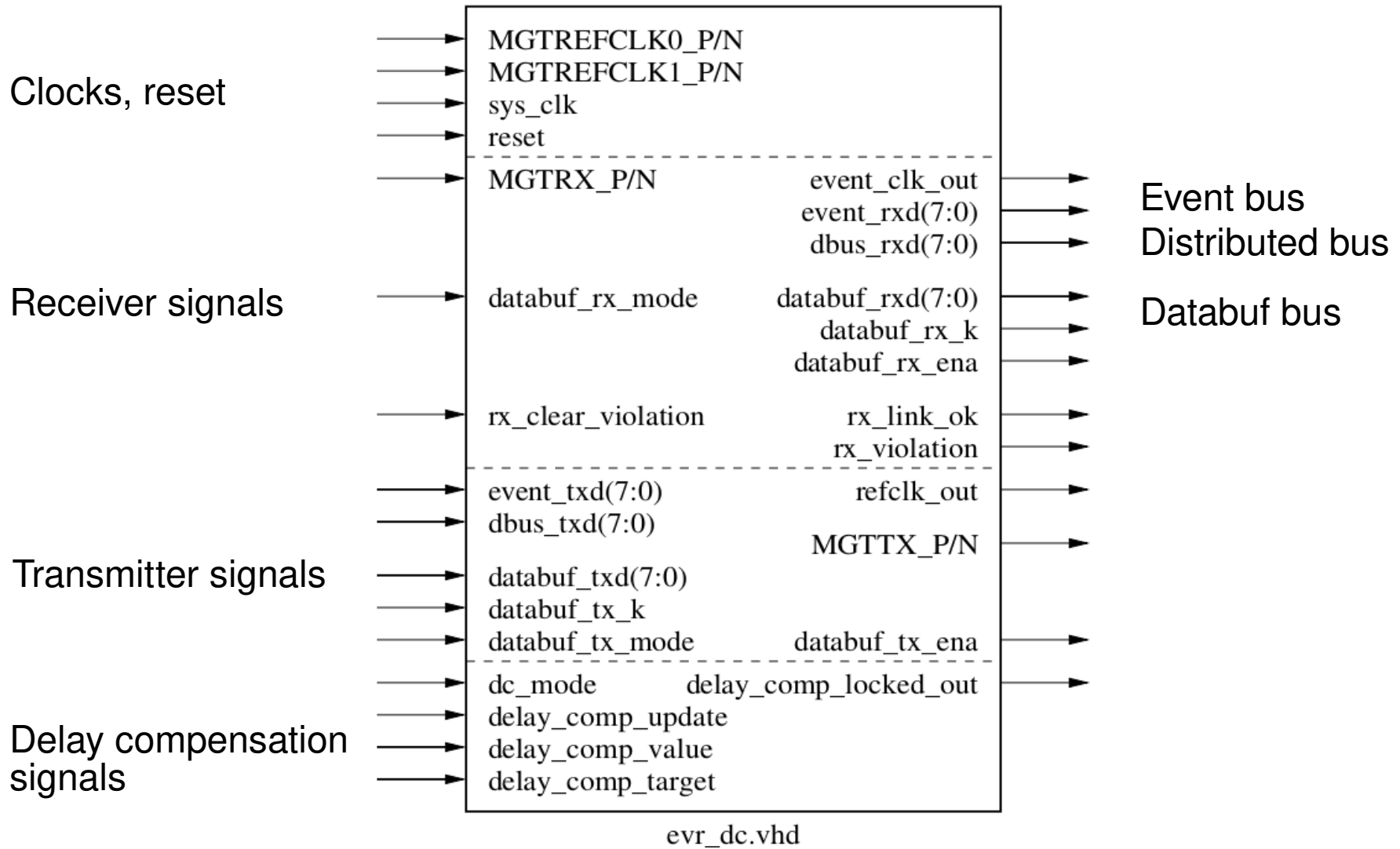
    databuf\_rx\_dc.vhd - Segmented data buffer receiver

        evr\_pkg.vhd

        buf\_bsram.vhd

zynq.xdc - Constraints file

# Open Source Event Receiver



# Minimum Configuration

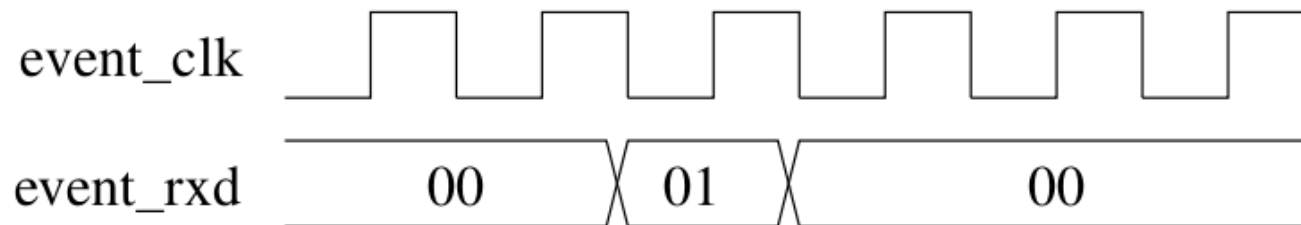
- Connect reference clock to one reference clock input MGTREFCLKx\_P/N
- Connect system clock to sys\_clk
- Connect transceiver receive signals to MGTRX\_P/N
- Tie databuf\_rx\_mode and databuf\_tx\_mode high '1'
- Tie dc\_mode low '0' (non-DC mode)
- Set delay\_comp\_target to a fixed value > 0x00050000, 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;
```

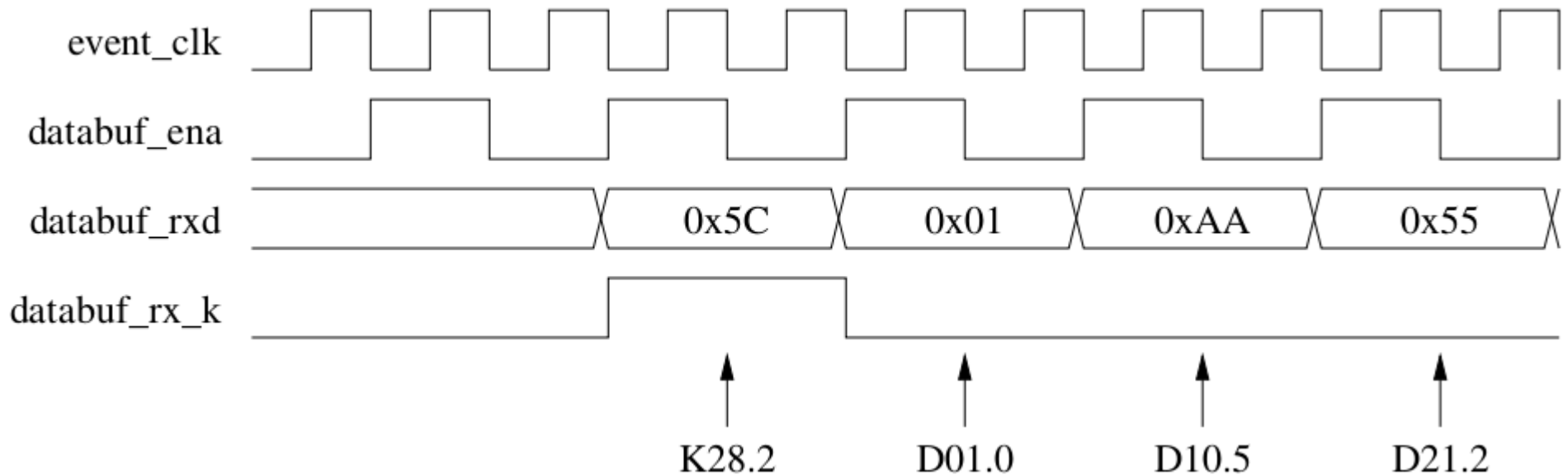


## Receiving Distributed Bus Bits

- 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

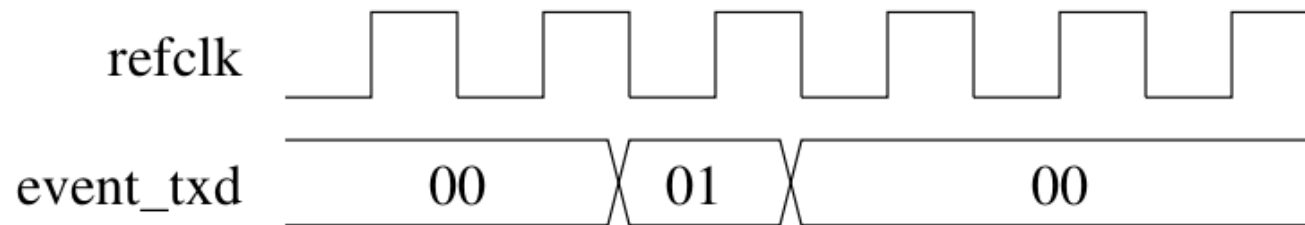
# Databuf bus

- 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

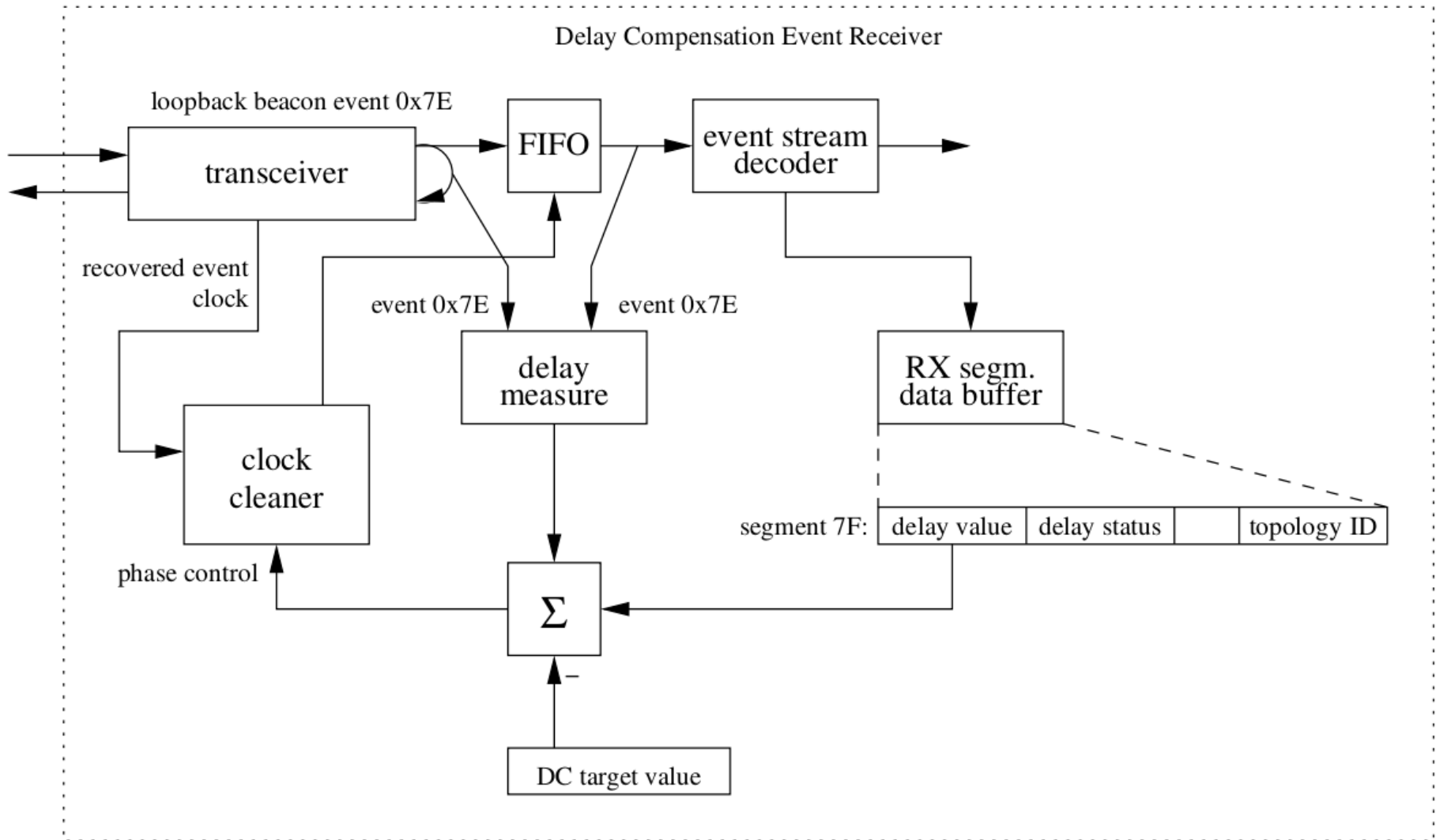


# Sending Events

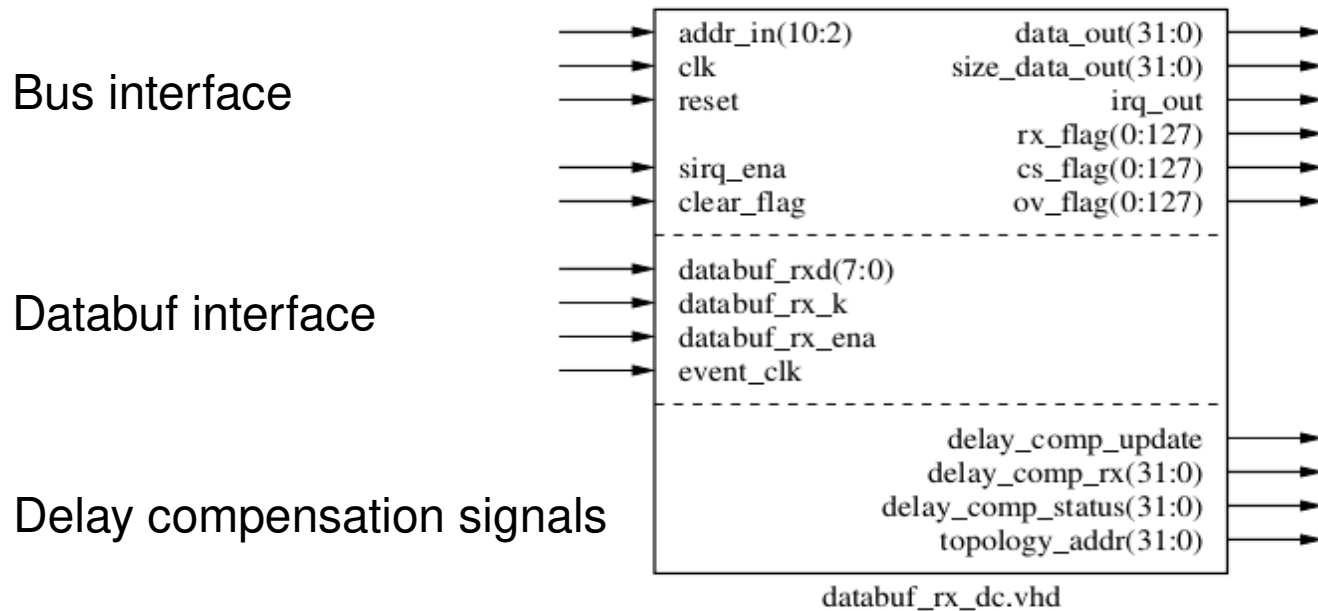
- Event codes are presented on `event_txd` in `refclk` clock domain
- When `event_txd` is not `0x00` the event code is sent out on the rising edge of `refclk`



# Open Source Event Receiver - DC mode



# Adding delay compensation



- `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

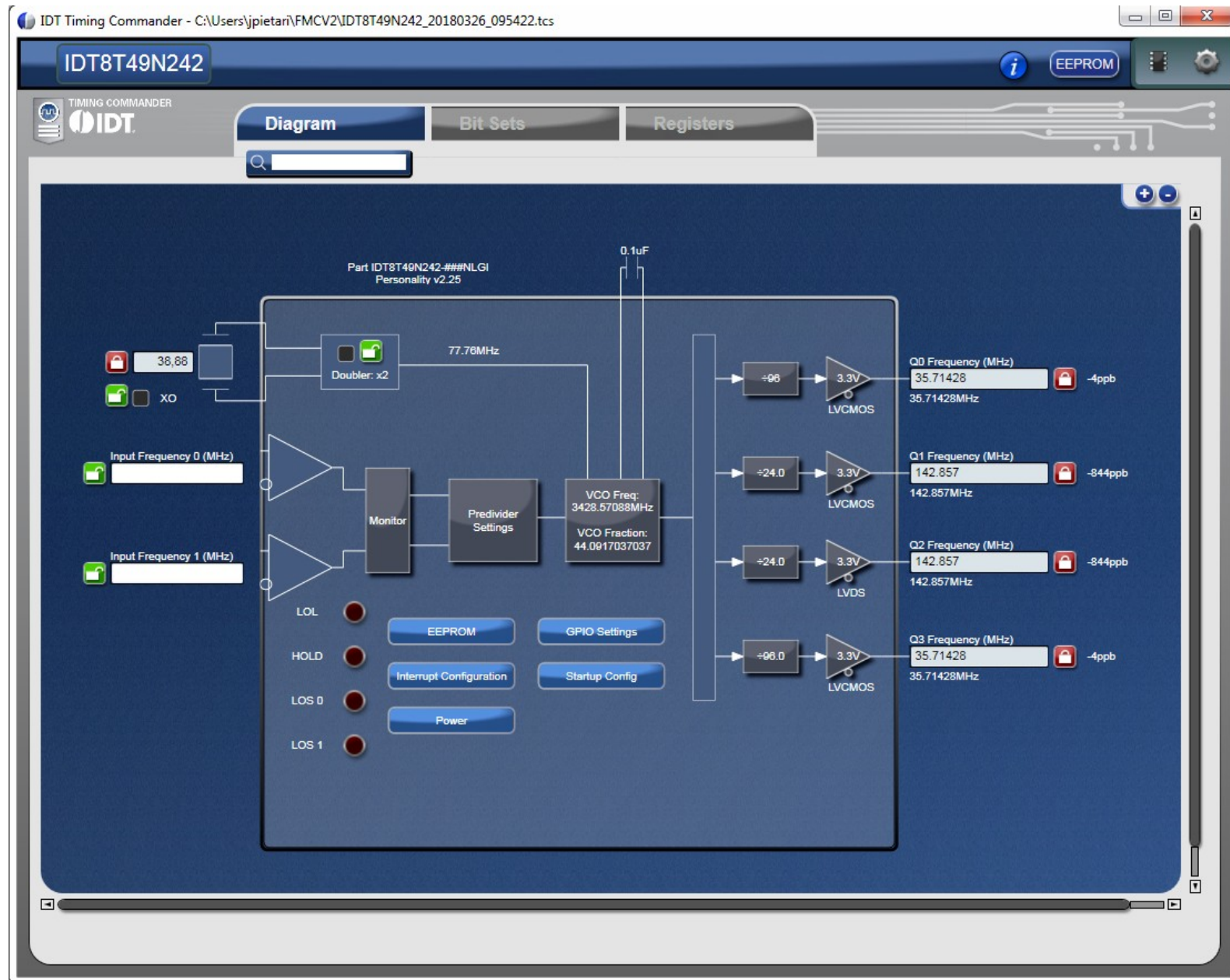
- LEDs
  - LED1 (D6), rx\_violation
  - LED2 (D7), link\_ok
  - LED3 (D8), flashed quickly on received 0x01 event code
  - LED4 (D9), flashes slowly
- Pushbuttons
  - SW1 (N), rx\_clear\_violation
  - SW2 (S), tx\_reset
  - SW3 (E), sys\_reset
- The event link can be looped back to itself (connect fiber 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<sup>th</sup> 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



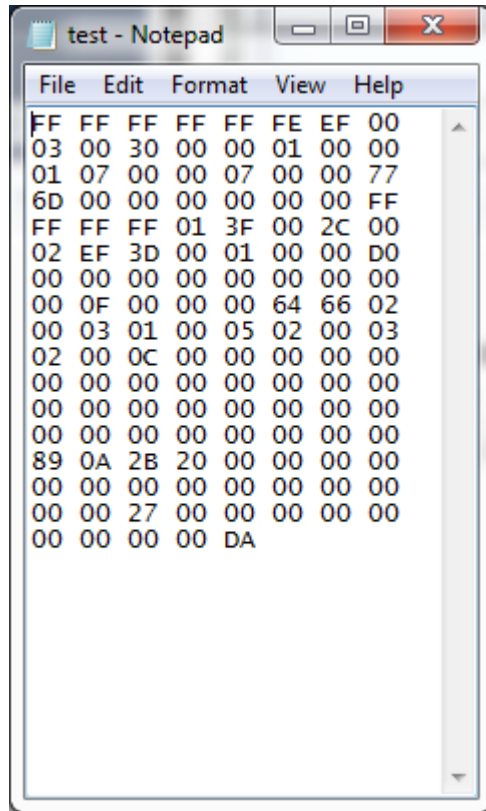
# Avnet PicoZed FMC Carrier V2 Board Reference Clock

- 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



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](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