# DIF HDL@llr

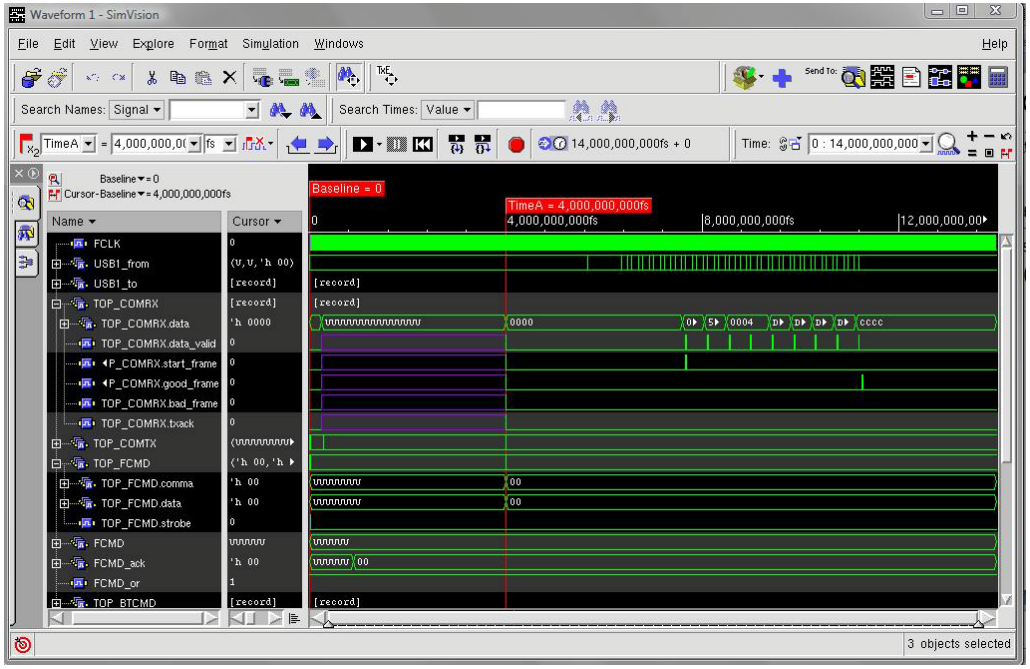
CALICE DIF task force meeting on EVO
Wednesday, October 14
14:30 CEST
Duration : 1h

# Use of « record » types

- Bundle of signals
  - Higher abstraction level
  - Makes the code clear
  - Ease the writing
  - Automatic propagation of modifications in the whole hierarchy
  - Ease common definitions of interfaces (put into a package)

```
type fcmd_t is record
  comma      : std_logic_vector(7 downto 0);
  data       : std_logic_vector(7 downto 0);
  strobe     : std_logic;
end record;
```

- Compatible with synthesis
- Compatible with simulation
- Limitations (?)
  - Ports of the same direction (all in or all out ?)
  - Unresolved type
- We (I) started to write code with “record” types



# « Record » types

```
BTCommandDecoder : entity work.DIF_BTCommandRegister(rtl)
generic map (
      G_rst_active => '0',
      G_en_active  => '1'
)
port map(
clk => Fclk, rst => '1', en => '1',
RX =>     TOP_downstream,
BTCMD =>  TOP_BTCMD,
BTCMDACK =>  TOP_BTCMDACK
);
```

```
-- add one pipe line reg to wait for comparators
RX_reg_p : process(clk)
begin
  IF clk'event and clk='1' then
      RX          <= padRX;
      --RX.data     <= padRX.data;
      --RX.data_valid <= padRX.data_valid;
      --RX.start_frame <= padRX.start_frame;
      --RX.good_frame <= padRX.good_frame;
      --RX.bad_frame  <= padRX.bad_frame;
  end if;--clk
end process;
```

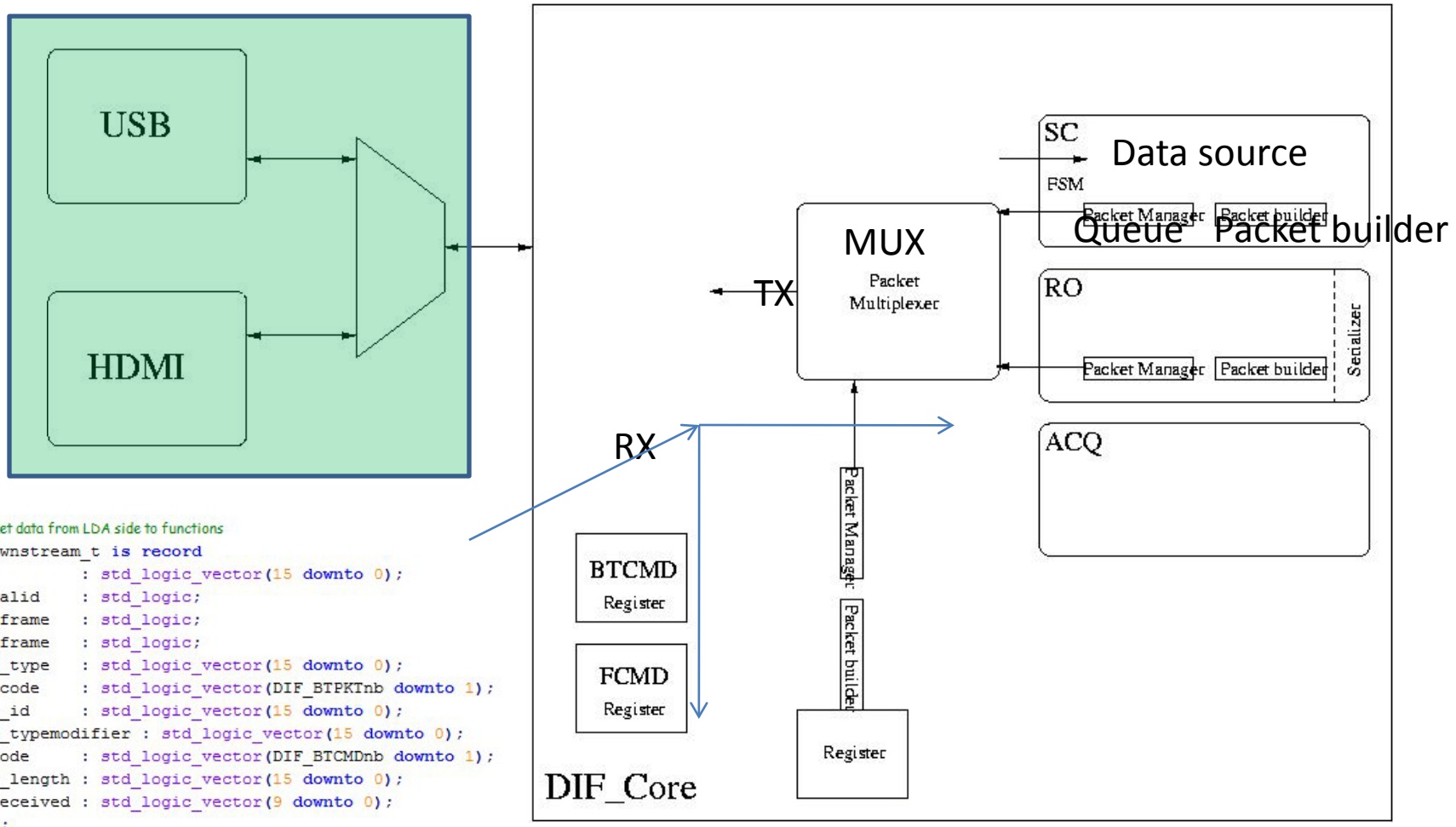
# HDL blocs (~tested)

- COM interface
  - USB
  - 8b/10b
  - Mux
  - Simulation back door
- DIF main FSM
- Fast command decoder & registers
- Fast command ack
- Block transfert command decoder & register
- RX packet decoder (FSM + registers)
- TX packet mux

# HDL blocs (~being developped)

- TX packet formatter
- Attempts of ROC functions
  - Data ser/des
  - Clk generator
  - Fsm & rams

# Standardized bloc diagram and internal interfaces ?




```
-- decoded packet data from LDA side to functions
type difdownstream_t is record
    data          : std_logic_vector(15 downto 0);
    data_valid    : std_logic;
    start_frame   : std_logic;
    error_frame   : std_logic;
    packet_type   : std_logic_vector(15 downto 0);
    PKT_decode    : std_logic_vector(DIF_BTPKTnb downto 1);
    packet_id     : std_logic_vector(15 downto 0);
    packet_typemodifier : std_logic_vector(15 downto 0);
    TM_decode     : std_logic_vector(DIF_BTCMDnb downto 1);
    packet_length : std_logic_vector(15 downto 0);
    word_received : std_logic_vector(9 downto 0);
end record;
```