Author : Bertrand CUZEAU – ALSE – http://www.alse-fr.com

Advanced Logic Synthesis for Electronics
www.ALSE-FR.com

# A.L.S.E. Application Note

# High-Speed LVDS Communication Demonstrator

## running on BeMicroMax10

## *Introduction*

This Application Note demonstrates how to implement high speed serial communication through LVDS links on low-cost FPGAs.

This demo has been developed for the very cheap BeMicroMax10 kit, which is unfortunately not available any more. Its cost was around $30, and it does expose LVDS pins on simple HE10 connectors. However, we had to use two pins on the edge connector to connect the LVDS clock input.

We chose to implement a 500Mb/s link : one clock lane at 50 MHz and one Data lane at 50x10=500 Mb/s.

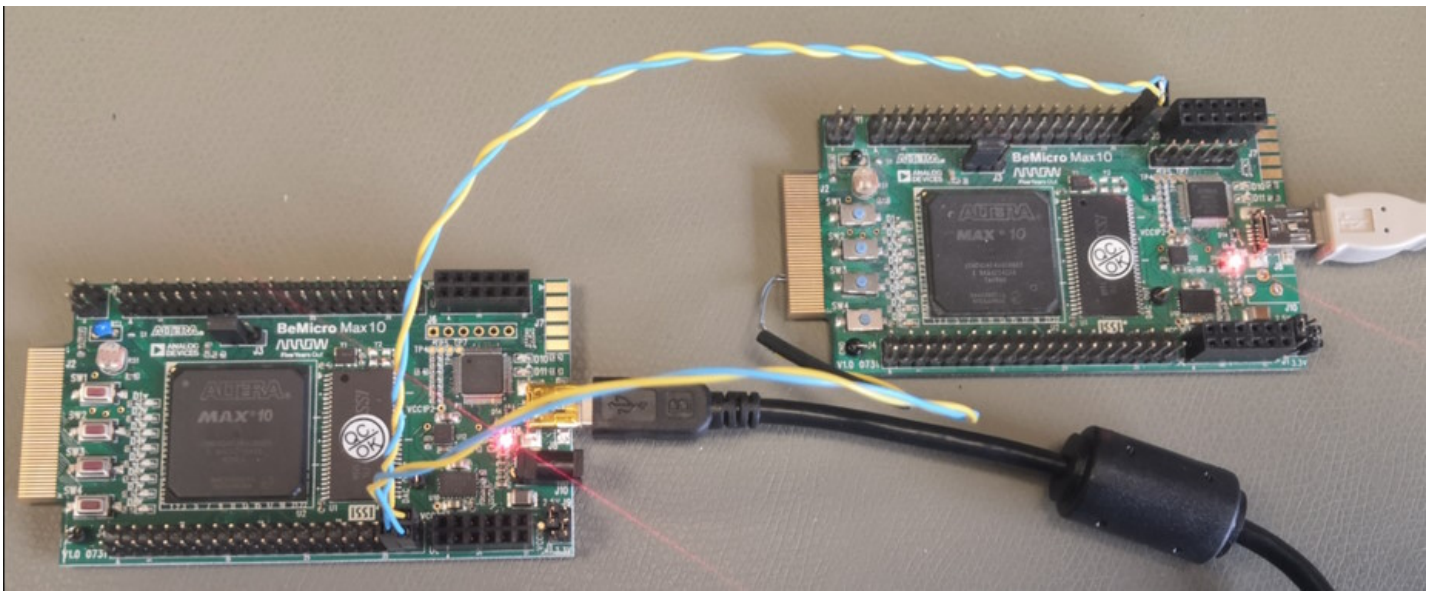
Even the slowest Max10 FPGA (55 nm) is capable of up to 640 Mb/s.

Most FPGAs from any vendor support high-speed LVDS communication.

The connectivity (low voltage differential signaling) is physically robust and easy to implement (the signal integrity challenge is low).

## *Demonstrator setup*

Here is the demonstrator, using one BeMicroMax10 as sender (left one) and one as receiver (right one).



## *Board to board connections*

### **Clock pair** (50MHz)

Sender : TX[0]=V17=J3-39=Clock

Receiver : RX[0]=Clock=N5=EG_P56  (EGP57=N4=Clk0n)
We had to use the edge connector for this pair. This is not very comfortable but it works.

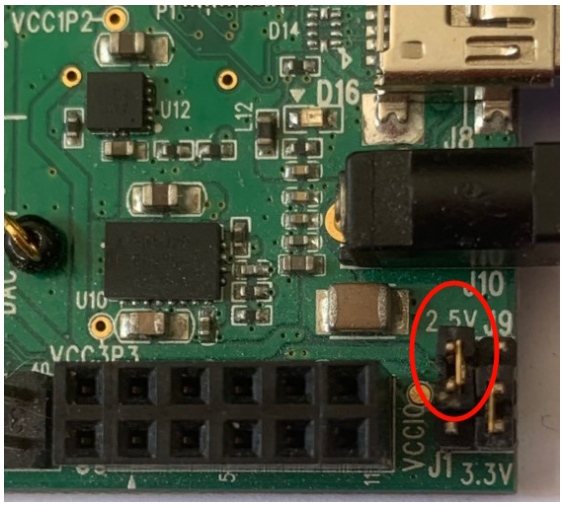
### **Data pair** (500Mbits/s)

Sender = TX[1]=V16=J4-37=Data

Receiver : RX[1]=E16=J3-37=Data

Note : the _N pins are assigned automatically by Quartus.

Important : the banks must be powered in 2.5V (not 3.3V).
The J1 jumper must be placed in upper position as shown below.




Simple twisted pairs have been used. The HE10 connectors are not ideal, nor are the routing of the differential pairs on the PCB.

Important : both pairs have been terminated by a 100 Ohms resistor close to the connector. This is not ideal, but it is sufficient to ensure error-free communication at 500 Mb/s.

## *Sender buttons*

SW2 when pushed force “11” on the two MS bits, indicating to the receiver that no Data is available.
This creates errors captured on the receiver side.

## *Receiver buttons & LEDs*

SW3 is the push button used to reset the Error counter & the persistent Error Flag LED

LEDS = Error Counter on LEDS[7:2]  -  Error_Latched  -  Error_pulsed

## *Communication format*

We chose a 10:1 format allowing to transmit 10 bits words per beat.

This format is very comfortable since it allows to use the 8 data bits with no protocol overhead.

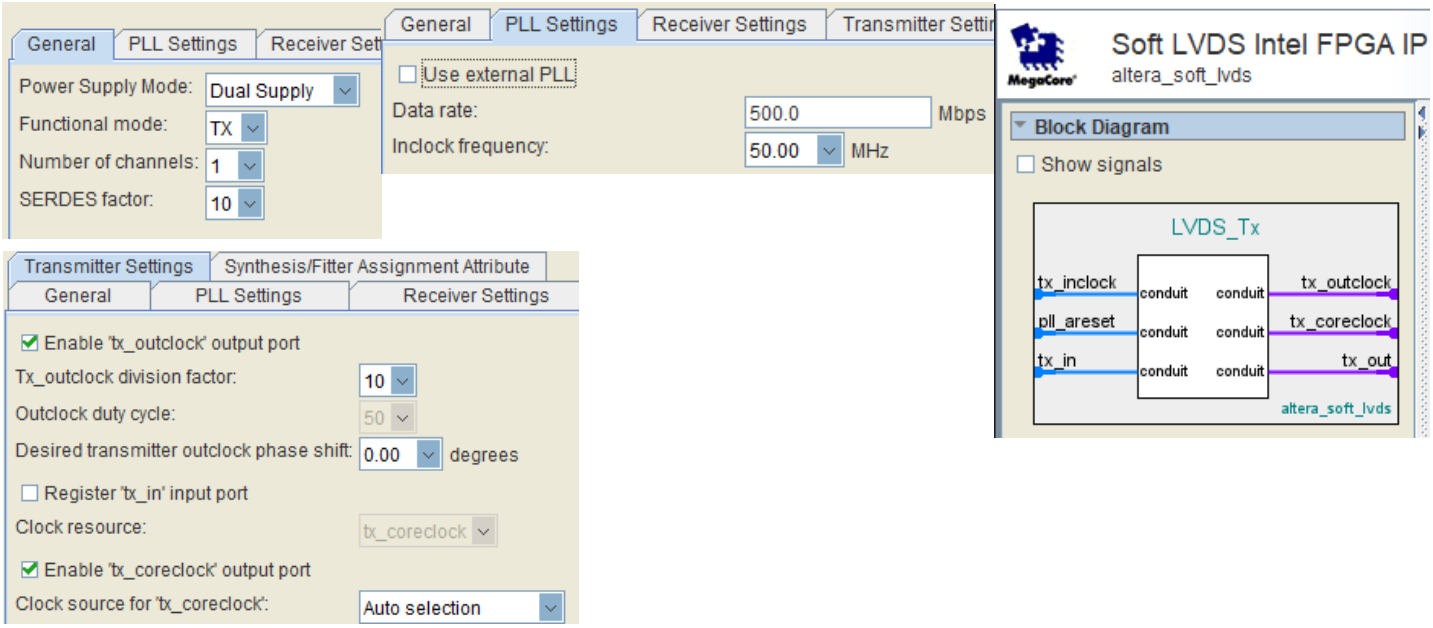
The 2 extra bits can for example define : Data Mode (10), Command Mode (01), Idle (11) and Invalid (00).

With the chosen settings, the effective data throughput is 400 Mbits/s.

Note that, by principle, the receiver does consume one PLL. Even the low cost and small Max10-8k LEs has two PLLs.

## *The Sender*

### LVDS Tx IP




It’s quite simple to generate the intended Serializer.

A PLL is automatically created and embedded in the IP block, but for the Tx function in a real design, it’s usually preferable to use a PLL *external* to the IP, which can be used for other purposes.
The “Use external PLL” option allows this.

For the LVDS receiver however, the required PLL can hardly be used for other purposes since the input clock may not always be present nor from a reliable frequency.

The settings of the LVDS Rx IP are very similar to the Tx indeed.

The LVDS Clock (tx_outclock) is at 50 MHz, as well as the tx_coreclock output.

## Sender Code (SystemVerilog)

The code for the Sender (Transmitter) is extremely simple :

```
/*
  ----------------------------
  TRANSMITTER
  LVDS 8:1 500Mb/s demo
  BeMicro Max10 kit
  ----------------------------
  (c) 2023 ALSE

  Lots of glitches on SW, this doesn't matter
  SW2 = push down to DISABLE Data mode & verification

*/

module top (
  input  logic       Clk,      // N14, 50 MHz
  input  logic [1:4] SW,       // active low push buttons with glitches
  output logic [7:0] LEDS,     // active low LEDs
  output logic [1:0] LVDS_TX_P // TX[1]=V16=J4-37=Data | TX[0]=V17=J3-39=Clock
);

logic TxClk;
logic [9:0] TxData; // Data counter
logic [1:4] SWr, SWrr;

// SWi resynch
always @ (posedge Clk) begin SWr <= SW; SWrr <= SWr; end

always @(posedge TxClk) begin
  TxData[7:0] <= TxData[7:0] + 1'b1;
  TxData[9:8] <= {1'b1,~SWrr[2]};
end

assign LEDS = {7'b1111111,SWrr[2]} ; // off except LEDS[0]=SW2

// LVDS Sender
// TxData is 10nnnnnnnn by default (enabled), must press SW2 to disable

LVDS_Tx iTx (
    .tx_inclock(Clk),
    .pll_areset(1'b0),
    .tx_in(TxData),
    .tx_coreclock(TxClk),
    .tx_outclock(LVDS_TX_P[0]),
    .tx_out(LVDS_TX_P[1])
  );

endmodule
```

We merely instantiate the LVDS transmit block (with the PLL embedded in this case).

The data sent is continuously incrementing. When the Data mode is disabled (by pressing SW2), the counting does continue but the receiver will ignore the data sent.

The sender logic is running from tx_coreclock (50 MHz).

## *LVDS Receiver code*

The code is also straightforward :

- Resynchronization (SW & Rst button)
- Tick generator
- Error monitoring and counter
- LVDS Rx IP instantiation
- Data checker

```
/*
  -------------------------------
  RECEIVER !
  LVDS 8:1 500Mb/s demo
  BeMicro Max10 kit
  -------------------------------
  (c) 2023 ALSE - Bertrand CUZEAU

  The Rx Clock pair must be attached to the
  80 pins edge connector EGP56-57

  Lots of glitches on SW, doesn't matter
  SW1  = General reset
  SW3  = push down to reset the Error counter & Error Flag
  LEDS = Error Counter [7:2] - Error_Latched - Error_pulsed
*/

module top (
  input  logic       Clk,      // N14, 50 MHz
  input  logic [1:4] SW,       // active low push buttons
  output logic [7:0] LEDS,     // active low LEDs
  input  logic [1:0] LVDS_RX_P // RX[1]=Data=E16=J3-37
);                             // RX[0]=Clock=N5=EG_P56  (Clk0n=N4=EGP57)

logic       RxClk;
logic [9:0] RxData;
logic [7:0] RxD, RxDr;
logic [1:4] SWr, SWrr;

// SWi resynch
always @ (posedge Clk) begin SWr <= SW; SWrr <= SWr; end

// Reset resynch (SW1)
logic RstRx;
RstSync r40 (~SWrr[1],RxClk,RstRx);

// clock divider & tick generation
logic [31:0] cnt = 0; // cnt[18] 100 Hz
logic cnt18r;
logic Tick;

always @(posedge RxClk) cnt <= cnt+1'b1;

`ifdef Simu
  always @(posedge RxClk) cnt18r <= cnt[9];  // 9 -> 20 us
  assign Tick = cnt18r ^ cnt[9];
`else
  always @(posedge RxClk) cnt18r <= cnt[18];  // 18 -> 10 ms
  assign Tick = cnt18r ^ cnt[18];
`endif

logic Err; // output (LED)
logic ErrM; // memorized
logic Trig, Trigr;
logic [2:0] Fcntr;  // max delay ~70 ms
logic [5:0] ErrCntr;
```

```
always @(posedge RxClk, posedge RstRx)
  if (RstRx) begin
    Fcntr <= 'b0;
    ErrCntr <= 'b0;
    Err <= 1'b0;
    ErrM <= 1'b0;
  end else begin
    // Monostable (Flasher)
    if (Tick) begin
      if (Fcntr != 0) begin
        Fcntr <= Fcntr-1'b1;
      end else begin
        Err <= 1'b0;
      end
    end
    // if multiple consecutive errors, only count the 1st
    if (Trig & !Trigr) begin
      Err <= 1'b1;
      ErrM <= 1'b1;
      Fcntr <= 'd7;
      if (ErrCntr!=63) begin
        ErrCntr <= ErrCntr + 1'b1;
      end
    end
      // Clear error on SW3 button pressed
    if (~SWrr[3]) begin
      ErrM <= 1'b0;
      ErrCntr <= 'b0;
    end
  end

  assign LEDS = ~{ErrCntr,ErrM,Err}; // LEDs are active low

// LVDS Receiver
LVDS_Rx iRx (
    .pll_areset (1'b0),
    .rx_outclock(RxClk),
    .rx_inclock (LVDS_RX_P[0]),
    .rx_in (LVDS_RX_P[1]),
    .rx_out (RxData)
  );

always @(posedge RxClk) begin
  if (RxData[9:8]==2'b10)  // Data mode
    RxD <= RxData[7:0];
    RxDr<= RxD;
end

// Data checker
assign Trig = ((RxD - RxDr) > 8'd1);
always @(posedge RxClk) Trigr <= Trig;

endmodule
```

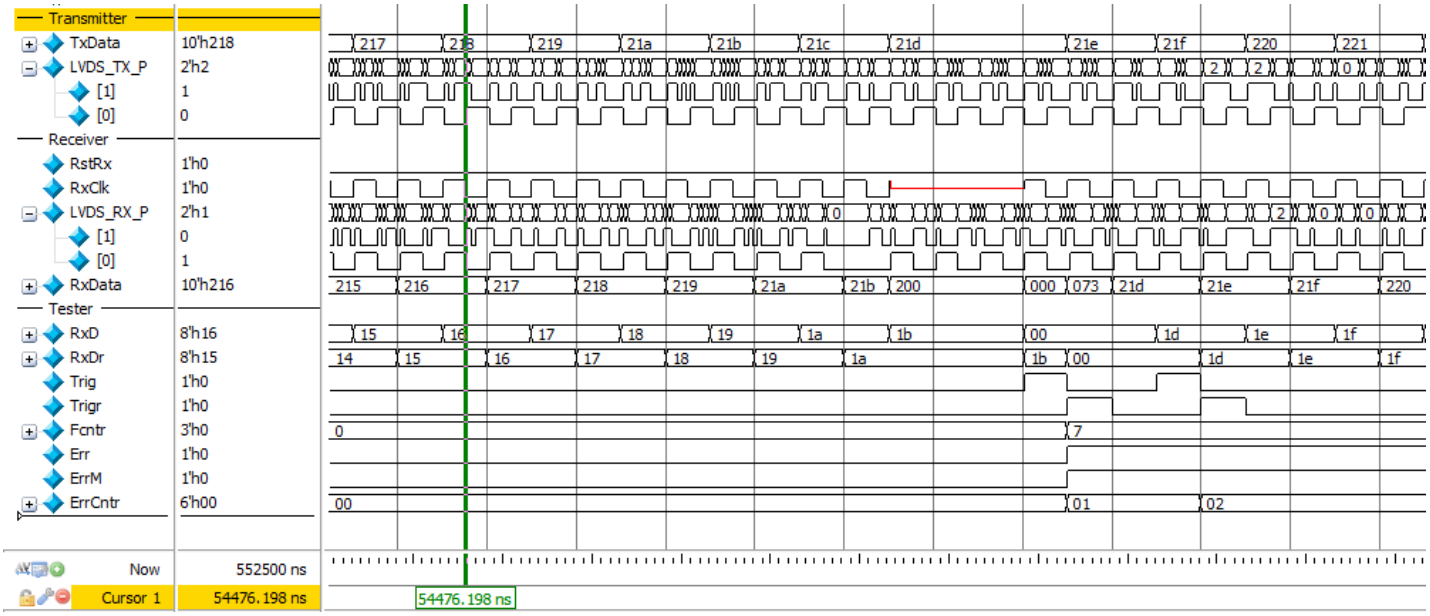
## SDC file

```
## BeMicro Max10 - "10M08SAE144C8GES"
# (c) ALSE

set_time_format -unit ns -decimal_places 3

create_clock -name Clk50 -period 20.000 [get_ports Clk]
derive_clock_uncertainty

# 500 Mb/s :
create_clock -name ClkLVDS -period 20.000 [get_ports {LVDS_RX_P[0]}]

derive_pll_clocks

set_false_path -to   [get_ports LED*]
set_false_path -from [get_ports SW*]

set_false_path -from [get_ports LVDS_RX*]
```

## *Simulation*

The design (Sender+Receiver) has been successfully simulated.

A physical fault has been injected on the clock lane.




We see that the link does quickly recover and only two data words are lost, captured by the checker.

(Note : this simulation was performed with 400Mbits/s settings)

## Scope capture

We have captured the LVDS clock and data signals :


The upper trace is the CLK_P wire (50 MHz clock).

The lower trace is the Data pair captured with an High-Speed active differential probe.
We see the 10 bits over the 20 ns period.
All words start by 1-0 on this capture.
The TUI (bit time) is -as expected- 2 ns.

## Test Results

This setup was easy to design and works flawlessly at 500Mb/s in spite of the sub-optimal conditions.

## Conclusion

This small project demonstrates that inter-FPGA high speed LVDS communication is easy to implement, efficient, and cost-effective.

Even in sub-optimal conditions (wiring, PCB routing, impedance termination, Engineering sample FPGA…), 500 Mb/s can be achieved easily and reliably.

This setup can be used as a sender (with another FPGA as receiver), or as a receiver (with another type of FPGA as sender).

Bertrand Cuzeau – Chief Technology Officer A.L.S.E
-=oOo=-