Key Design Features

- Technology independent soft IP Core for FPGA, SoC and ASIC
- Supplied as human readable VHDL (or Verilog) source code
- Separate LVDS Transmitter / Receiver (SERDES) pair
- Up to 8 serial LVDS data lanes + LVDS clock
- Fully configurable clocking
- Generic parallel data width up to 128 bits wide
- Generic parallel-to-serial mux ratio up to 16:1
- Data rates of up to 1 Gbits per lane
- Integrated asynchronous FIFOs with underflow / overflow detection
- Bitwise data alignment at the receiver
- No receiver source clock required
- Compatible with a wide range of industry standard protocols including: Channel Link®, Camera Link®, FPD Link®, FlatLink® etc.
- Robust and simple to implement using cheap twisted pair cable (e.g. Cat 5 Ethernet)

Example Applications

- High bandwidth SERDES interfaces
- Serialization of wide buses e.g. 'virtual' ribbon cable
- Direct replacement for many commercial LVDS / SERDES ICs
- Data streaming interfaces over cable or twisted pair over longer distances

Generic Parameters

Generic name	*Description*	*Type*	*Valid range*
dw	Parallel data width	integer	2 ≤ dw ≤ 128
ratio	Parallel-to-serial multiplexer ratio	integer	2 ≤ ratio ≤ 16
duty	Transmitter clock duty cycle setting	integer	0 < duty < ratio
skew	Transmitter clock skew setting	integer	0 ≤ skew ≤ ratio
lanes	Number of serial data lanes	integer	dw / ratio (8 max)
direction	Serialization / Deserialization direction	integer	0: forward 1: backward
polarity	Receiver clock sampling edge	integer	0: -ve edge 1: +ve edge

Block Diagram

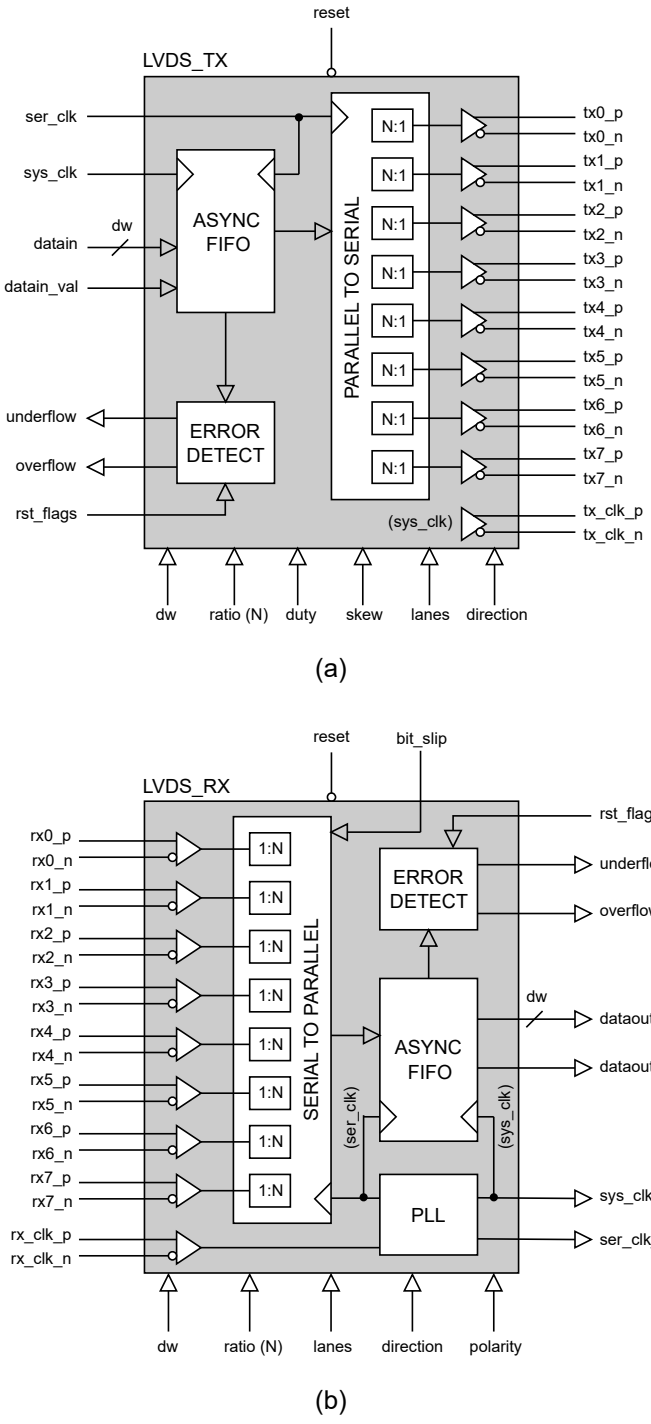

Figure 1: LVDS (SERDES) Transmitter (a) and Receiver (b) basic architecture

Pin-out Description

LVDS TRANSMITTER

Pin name	I/O	Description	Active state
sys_clk	in	System clock (Synchronous with parallel input data)	rising edge
ser_clk	in	Serial clock (= sys_clk x ratio)	rising edge
reset	in	Asynchronous reset	low
underflow	out	Error flag indicates starvation of data	high (sticky until reset)
overflow	out	Error flag indicates a surplus of data	high (sticky until reset)
rst_flags	in	Reset the underflow / overflow error flags	high
datain [dw-1:0]	in	Parallel input data	data
datain_val	in	Parallel input data valid	high
txN_p (max 8)	out	Positive Tx strobe serial data lane 'N'	LVDS
txN_n (max 8)	out	Negative Tx strobe serial data lane 'N'	LVDS
tx_clk_p	out	Positive Tx clock strobe	LVDS
tx_clk_n	out	Negative Tx clock strobe	LVDS

LVDS RECEIVER

Pin name	I/O	Description	Active state
sys_clk_f	out	Recovered system clock (Synchronous with parallel output data)	rising edge
ser_clk_f	out	Recovered Serial clock (= sys_clk_f x ratio)	rising edge
sys_rst_f	out	Resynchronized system reset	low
reset	in	Asynchronous reset	low
underflow	out	Error flag indicates starvation of data	high (sticky until reset)
overflow	out	Error flag indicates a surplus of data	high (sticky until reset)
rst_flags	in	Reset the underflow / overflow error flags	high
bit_slip	in	Causes parallel data output word to be barrel-shifted by one bit. (Used to align output data)	rising edge
dataout [dw-1:0]	out	Parallel output data	data
dataout_val	out	Parallel output data valid	high
rxN_p (max 8)	in	Positive Rx strobe serial data lane 'N'	LVDS
rxN_n (max 8)	in	Negative Rx strobe serial data lane 'N'	LVDS
rx_clk_p	in	Positive Rx clock strobe	LVDS
rx_clk_n	in	Negative Rx clock strobe	LVDS

General Description

The LVDS_SERDES IP Core is a high-speed LVDS transmitter / receiver pair suitable for a wide range of serial interface applications. The design is comprised of an independent transmitter and receiver that may be used separately or together as a single transceiver.

The transceiver can accept parallel data widths of up to 128-bits and features a user-defined multiplexer ratio. By modifying the generic parameters, *dw*, *ratio, duty, skew, lanes* and *direction,* the transceiver can be made compatible with a wide range of third-party LVDS devices such as those from National Semiconductor®, TI®, Thine® and Maxim®.

In total, the transceiver can support up to 8 serial data lanes - each data lane typically handling rates of between 500 Mbits/s and 1Gbits/s. The maximum data rate attained will be dependent on a wide range of factors such as: cable type, cable length, board layout, and the specification of the LVDS buffers. As a general rule, data rates of 350 Mbits/s per lane can be easily achieved on even the most basic FPGA platforms.

In addition to the 8 data lanes, a single clock lane is provided for synchronizing the data between the transmitter and receiver. Figure 1 shows the basic architecture of the transmitter and receiver pair. The following sections explain the transmitter and receiver functionality in more detail.

LVDS Transmitter

The transmitter is responsible for serializing the parallel input data into separate data lanes. The input data is partitioned into 'N' groups, where the width of each group is defined by the generic parameter *ratio*. As an example, consider a parallel data width of 21-bits and a mux ratio of 7. The resulting architecture would have 3 data lanes in an arrangement like that shown in Figure 2 below:

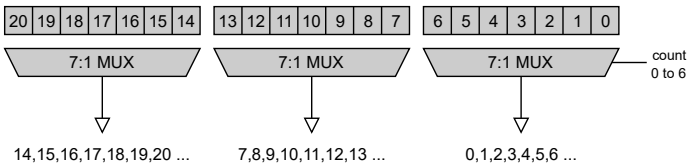

Figure 2: Multiplexer arrangement for a data width of 21-bits and a ratio of 7

The output order of the bits within each multiplexer is controlled by the generic parameter *direction*. With direction set to '1' then the serial bits are multiplexed in the order 0, 1, 2, … etc. When direction is '0' then the order is reversed. The direction parameter is provided for compatibility with various third party SERDES solutions and ICs.

The transmitter requires two separate clocks for correct operation. The signal *sys_clk* is a system clock that is synchronous with the input data. The signal *ser_clk* is the serial clock. The system clock and serial clock do not need to be phase-aligned, but the serial clock must be an exact integer multiple of the system clock with the relationship:

$$\text{ser\_clk} = \text{sys\_clk} * \textit{ratio}$$

After system reset, transmission of data begins on a rising clock-edge of *sys_clk* when *datain_val* is asserted high. The serialization process then begins with parallel data words being read on consecutive system clock cycles.

During operation, the asynchronous FIFO detects the data rate into and out of the transmitter. If at any point the FIFO overflows or is starved of data, then the respective error flags *overflow* or *underflow* are asserted. These flags may also be asserted if the relationship between the system clock and serial clock is not maintained.

Note also that asserting the bit-slip command may cause the error flags to be asserted. For this reason, the flags should only be observed during normal operation when any data-alignment process has been completed. Once set, these flags remain high until a specific reset using the *rst_flags* signal. A system reset will also reset these flags, but any data alignment at the receiver may be lost.

LVDS Transmitter clocking

The LVDS transmitter clock configuration is specified using the generic parameters 'duty' and 'skew'. The parameter *duty* specifies the number of serial clock cycles that the transmitter clock is in the active low state. An example of this is shown in figure 3 below. The *skew* parameter permits the user to skew the transmitter clock (in serial clock cycles) relative to the LVDS data. By adjusting these parameters, the IP Core may be used to duplicate the clocking behaviour of most commercial LVDS ICs.

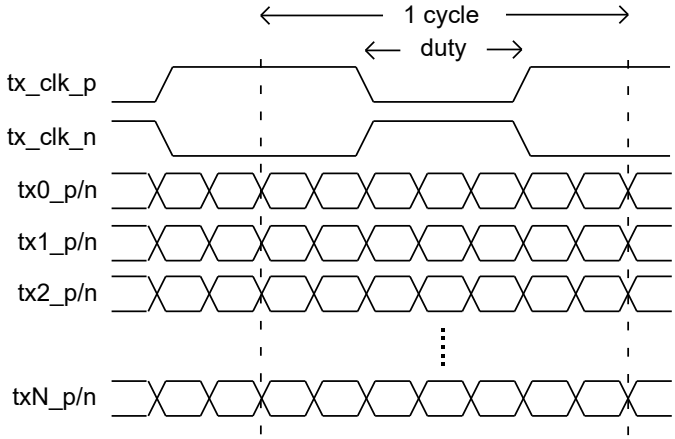

Figure 3: Transmitter clock specification showing a duty cycle setting of 3 serial clock cycles

LVDS Receiver

The receiver performs the reciprocal operation to the transmitter and is responsible for de-serialization of the serial input data. Clock recovery and serial clock generation is performed by an internal PLL or a dedicated MMCM clock management component[1]. The point at which the data is sampled (point within the data 'eye') may be controlled by the generic parameter *polarity*.

Setting polarity to '0' results in the data being sampled on a falling clock-edge close to the centre of the eye. Setting polarity to '1' results in data being sampled on a rising clock edge. The best setting will depend on the implementation.

After system reset, the de-serialization process begins with parallel data words being output on consecutive cycles of the system clock. Data is valid from the point at which the signal *dataout_val* is asserted high.

As with the transmitter, the asynchronous FIFO monitors the data rate into and out of the receiver. If at any point, the clocks become out of sync, then the respective error flags *overflow* or *underflow* are asserted.

Data alignment at the Receiver

For most situations it's not always practical to perfectly align the parallel data at the transmitter with the parallel data at the receiver. This is because after reset, the input serial data bits could be at any point within an N-bit word. In order to correct this, the receiver employs a *bit_slip* signal that allows the output data word to be barrel-shifted by one bit. The bit slip signal is active on a rising-edge.

For instance, consider the case where the 32-bit pattern '0x44440000' is transmitted with a 4:1 mux ratio. At the receiver end, the 32-bit output word is observed as 0x2222000. In order to align the word correctly, the *bit_slip* signal must be toggled until the the correct output is observed.

This is shown graphically in Figure 4 below:

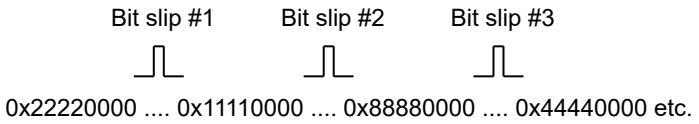

Figure 4: Receiver 'bit_slip' function

Data alignment at the receiver can normally be done quite simply by using a state machine that monitors the receiver output for a special character or pattern. For example, with digital video, this could be a Start of Active Video (SAV) code or an End of Active Video (EAV) code or some combination of the two.

The state machine would monitor the output for these codes and periodically assert *bit_slip* until the codes are detected and the output data is properly aligned.

An example bit-slip implementation is provided in the VHDL source file: *lvds_data_align.vhd*. This may be used for reference in order to align to a specific data pattern.

1 Most EDA tools (e.g. AMD / Vivado) feature applications that allow easy generation of PLL or MMCM components with the desired parameters.

Functional Timing

Figure 5 shows the serialization of a 32-bit data word with a mux ratio of 4:1. In this example, all 8 serial data lanes are being used. Note that the frequency of the serial clock is exactly 4 times the frequency of the system clock. The de-serialization process at the receiver has exactly the same timing relationship - but performs the inverse operation.

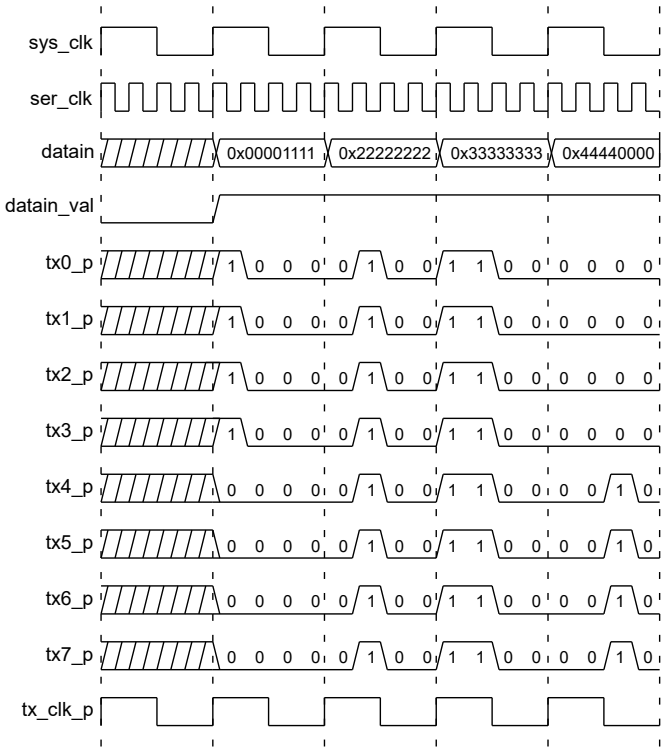

Figure 5: Example functional timing for the serialization of a 32-bit data word with a mux ratio of 4:1 and all 8 serial data lanes being utilized. The direction is set to '1'.

Source File Description

All source files are provided as text files coded in VHDL. The following table gives a brief description of each file.

Source file	*Description*
pipeline_reg.vhd	Pipeline register
fifo_async.vhd	Asynchronous FIFO
lvds_obuf.vhd	Differential output buffer
lvds_oclk.vhd	Differential output clock generator
lvds_ibuf.vhd	Differential input buffer
lvds_iclk.vhd	Differential input clock buffer
lvds_pll.vhd	PLL for de-skew and serial clock gen
lvds_flow_error.vhd	Overflow / underflow error detector
lvds_deserializer.vhd	Instantiates N x 1:N deserializer
lvds_deserializer_n.vhd	1:N deserializer
lvds_serializer.vhd	Instantiates N x N:1 serializer
lvds_serializer_n.vhd	N:1 serializer
lvds_data_align.vhd	Example bit-slip controller for alignment
lvds_tx.vhd	Top-level LVDS transmitter component
lvds_rx.vhd	Top-level LVDS receiver component
lvds_serdes_bench.vhd	Top-level test bench

[Note: The components lvds_obuf.vhd, lvds_ibuf.vhd, lvds_oclk.vhd, lvds_iclk.vhd and lvds_pll.vhd are technology-specific components. These components must be changed for equivalent parts for correct implementation. Please contact Zipcores if further assistance is needed]

Functional Testing

An example test bench is provided for use in a suitable hardware simulator. The compilation order of the source code is the same order as the source code file description (above).

The test bench instantiates the transmitter and receiver top-level components in series such that the output of the transmitter feeds directly to the input of the receiver. The basic simulation setup is as follows:

Data in → Capture input → LVDS Tx → LVDS Rx → Capture output

The generic parameters *dw*, *ratio*,, *duty*, *skew*, *lanes, direction* and *polarity* have been set to 32, 4, 2, 0, 8, 1 and 1 respectively for the test. The user is free to modify these parameters as required to suit their specific test environment.

The simulation must be run for at least 1 ms during which time the LVDS transmitter is fed a random sequence of 32-bit words. Two output text files are generated during the course of the simulation. These files are 'lvds_in.txt' and 'lvds_out.txt' and contain a list of data words captured at the inputs and outputs of the transmitter and receiver. The equivalence of these files proves the correct operation of the test.

Note that at the start of the test, the 'bit_slip' signal is asserted various times in order to align the data correctly at the receiver. If the generic settings are changed, then the user may have to modify the number of bit-slip operations accordingly.

Development Board Testing

The LVDS (SERDES) IP Core was tested in a live demo using the Zipcores HD-video development board. The devboard is based on a AMD / Xilinx® Spartan FPGA and features a number of general purpose LVDS I/O pins.

An LVDS serial link was used to transmit a WXGA (1280x800) 24-bit RGB video signal to a Sharp® LQ101K1LY04 LCD display. The connections were set up for the Thine® THC63LVDF84B LVDS receiver IC. Figures 6 and 7 show photos of the general demo setup.

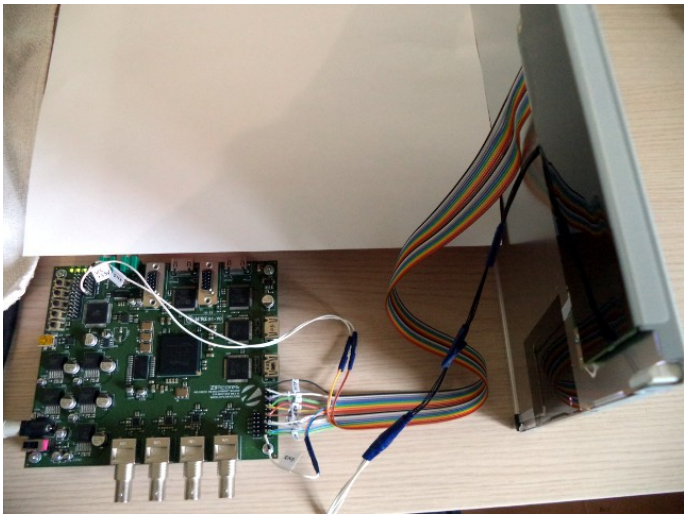

Figure 6: Demo setup with the Zipcores HD-Video board

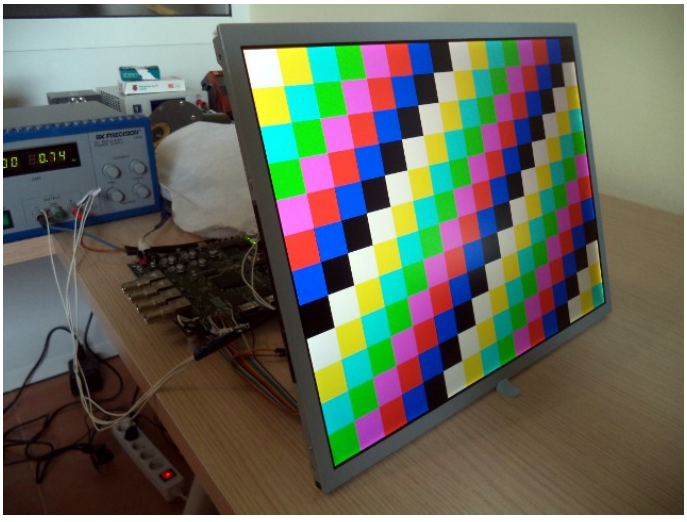

Figure 7: LVDS demo showing Sharp LCD (WXGA) test pattern display

Synthesis and Implementation

The files required for synthesis and the design hierarchy is shown below:

Transmitter top-level component:

- lvds_tx.vhd
  - lvds_serializer.vhd
    - lvds_serializer_n.vhd
    - lvds_flow_error.vhd
    - fifo_async.vhd
      - pipeline_reg.vhd
  - lvds_obuf.vhd
  - lvds_oclk.vhd

Receiver top-level component:

- lvds_rx.vhd
  - lvds_deserializer.vhd
    - lvds_deserializer_n.vhd
    - lvds_flow_error.vhd
    - fifo_async.vhd
      - pipeline_reg.vhd
  - lvds_pll.vhd
  - lvds_ibuf.vhd
  - lvds_iclk.vhd

The LVDS SERDES IP Core is technology independent with the exception of the differential LVDS I/O buffers and PLL component which must be specific to the FPGA, SoC or ASIC process being used. As a benchmark, synthesis results have been provided for the AMD / Xilinx® 7-series FPGAs. Synthesis results for other FPGAs and technologies can be provided on request.

One recommendation is to ensure that the I/O registers are placed in the pads of the target device. This may be specified as an additional attribute in the RTL source code or specified in the constraints file - for instance the 'xdc' or 'sdc' file in the synthesis tool[2]. Placing the inputs in the I/O of the device will ensure more reliable data capture and timing results.

Finally, it's also recommended that the user place the LVDS input and output pins in a localized area, that is, not spread out around the die. This will reduce timing skew between the input and output data.

Trial synthesis results are shown with the generic parameters set as follows: dw = 56, ratio = 7, duty = 3, skew = 2, lanes = 8, direction = 1, polarity = 1.

The resource usage is specified after place and route and is listed for the combined transmitter and receiver components in a series configuration.

2 Example constraints files may be provided on request according to the chosen synthesis tool.

AMD / XILINX® 7-SERIES FPGAS

Resource type	***A-7***	***K-7***	***V-7***	***V-US+***
Slice Register	266	266	266	266
Slice LUTs	169	169	169	166
Block RAM	0	0	0	0
DSP	0	0	0	0
Occupied Slices	75	88	87	51 (CLB)
System clk freq. (approx.)	70 MHz	80 MHz	90 MHz	100 MHz

Revision History

Revision	***Change description***	***Date***
1.0	Initial revision	12/09/2010
1.1	Added detailed bit_slip section	02/10/2010
1.2	Added development board test setup descriptions	07/01/2011
1.3	New generic parameter 'direction' for compatibility with various commercial LVDS ICs. Added signal to reset error flags	15/05/2011
1.4	Added new generic parameters: clock 'duty' and 'skew' to give further compatibility with commercial LVDS ICs.	09/02/2015
1.5	Updated everything to the latest AMD / Xilinx® 7-series components. Revised the datasheet and synthesis results.	22/01/2025