

1 Introduction

This application note describes the methods to use HME HR Series FPGA devices for high-performance LVDS interfaces. LVDS is a signaling standard that provides high-speed data transfers. HR series FPGA devices offer easy integration of LVDS interfaces at speeds up to 800 Mbps for the receiver transmitter. This application note also includes step-by-step design flow and interface guidelines. With simple settings, user can easily add the LVDS to the design through IP wizard in Fuxi software.

2 HME HR FPGA LVDS Overview

The HR FPGA LVDS IO supports the following features:

- Schmitt trigger input;
- Supports single or double data rate architecture
- Supports 1.5v/1.8v/2.5v/3.3v LVCMOS when using as single-ended IO
- Supports LVDS 2.5v and 1.8v
- Programmable driving strength
- Support pull-up, pull-down and keep function
- Performance up to 800Mbps

For the LVDS pair number of different HR FPGA product and package, please refer to:

[HME-HR_Family_Data_Sheet_EN.pdf](#)

The following tables and diagram show the DC spec of HR FPGA LVDS IO.

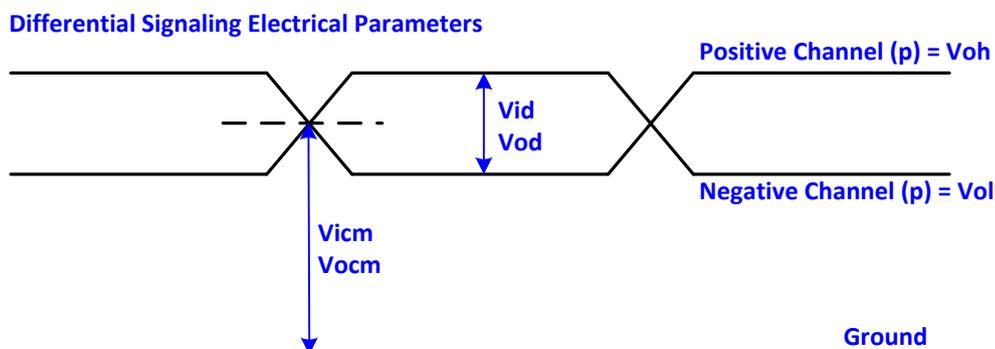


Figure 1 LVDS differential signaling electrical parameters

Table 1 input DC Spec of differential I/O standard

I/O Standard	Vccio (V)			Vid (V)			Vicm (V)			Vinp (input voltage)	
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Max
LVDS	2.375	2.5	2.625	0.25	0.35	0.45	Vccio/2-0.3	Vccio/2	Vccio/2+0.3	0	2.5
subLVDS	1.71	1.8	1.89	0.1	0.15	0.2	Vccio/2-0.25	Vccio/2	Vccio/2+0.25	0	1.8

Table 2 output DC Spec of differential I/O standard

I/O Standard	Vod (mV)			Delta(Vod) (mV)		Vocm (V)			Voh (V)		Vol (V)	
	Min	Typ	Max	Min	Max	Min	Typ	Max	Min	Max	Min	Max
LVDS		300					1.25					
subLVDS		350					0.9					

3 LVDS Usage

3.1 Basic design flow

The basic design flow for LVDS related application based on HR FPGA device is shown as below:

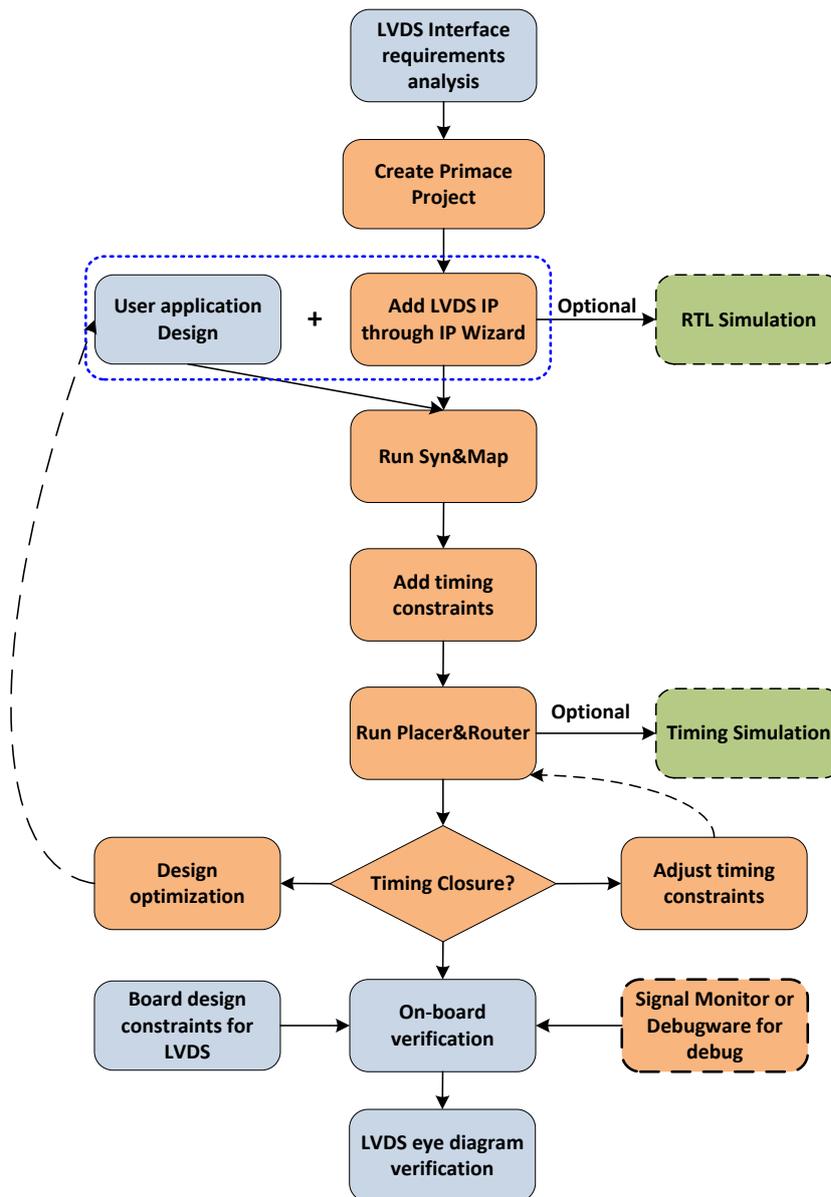


Figure 2 Basic LVDS application design flow based on HR FPGA

The basic LVDS application design flow including the following steps:

Step1 LVDS interface requirements analysis, user should decide the number of LVDS channels, deserialization factors, the LVDS data rates per channel according to application requirements.

Step2 Create Fuxi project and add LVDS IP to the design through IP Wizard and finish user application design

Step3 Run through Fuxi flow.

In this step, user need to first add timing constraints for the user logic clock and LVDS Tx/Rx clock after finish synthesis and mapping. Then implement P&R (Placement &Router) and

verify the timing through timing analysis. If there are timing violations, user can adjust the timing constraints and run P&R again or optimize the RTL design until there are no timing violation.

Fuxi also support lunch third party simulation tool Modelsim to do the RTL and timing simulation. User can run simulation before on board testing.

Step4 Board level design constraints. Based on HME HR FPGA to implement LVDS function, there are some board level design requirements, will be introduced in the following chapters.

Step5 Generate the configure bitstream, download and do on-board verification. Fuxi provides two kinds of debug tools: Debugware and Signal monitor to help user on chip debug. User can also directly use the oscilloscope to check the eye diagram of LVDS.

3.2 Factors need to be considered before LVDS interface design

Take LVDS transmission as example, before the LVDS interface design, the flowing factors need to be considered: Tx data width, serialization factor, user logic clock frequency, LVDS Tx clock frequency, number of LVDS channel, show in the diagram below. For LVDS reception, it is the same as transmission.

For example, the user logic has 28bit data, 60MHz, need to send out through LVDS interface, the relation between the factors list up are shown in table below.

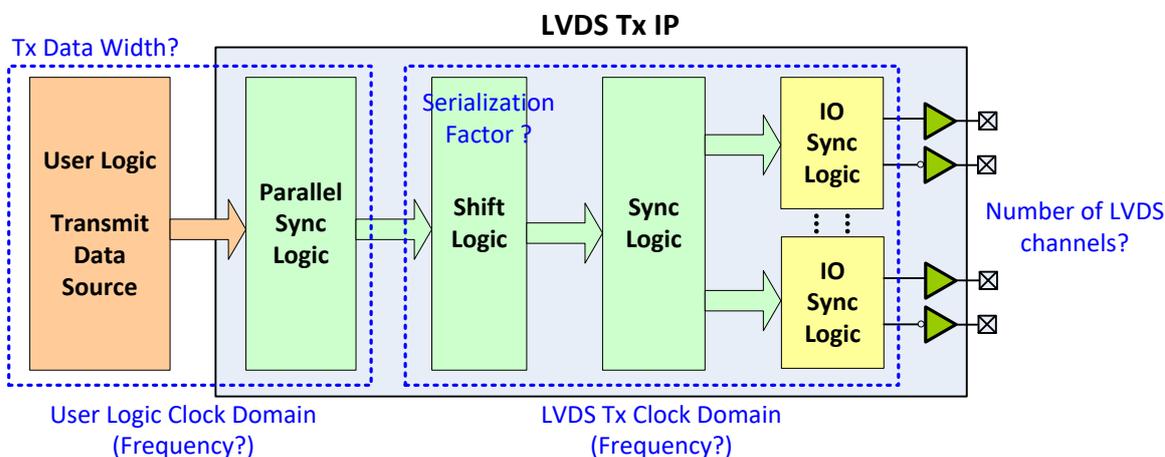


Figure 3 Factors need to be considered before LVDS interface design

Table 3 The relation between LVDS interface design related factors

Serialization Factor	Number of channels	User Logic Frequency	LVDS Tx Clock Frequency	Data rates per channel
x2	14	60MHz	60MHz	120Mbps
x4	7	60MHz	120MHz	240Mbps
x7	4	60MHz	210MHz	420Mbps

NOTE:

- ☑ LVDS IP support x2, x4, x7, x8 serialization factor
- ☑ The LVDS IP always use the DDR(Double Data Rate) mode to sample and output the data, so:

$$\text{LVDS Tx Clock Frequency} = (\text{Serialization factor}/2) * \text{User Logic Frequency}$$

For most of the applications, we can have a specific requirement on the number of LVDS channels or transmission data width. Then we can based on this factor and adjust other factors to satisfy the final application requirements. The frequency of user logic and LVDS Tx clock frequency are two factors need to add timing constraints and verify.

3.3 Add LVDS IP into design through Fuxi IP wizard

Fuxi provides the complete soft LVDS transmission and reception IP solution to speed up user design. The soft LVDS IP supports the following features:

- Both LVDS transmitter (serializer) and receiver (deserializer)
- At speeds up to 800Mbps for transmitter and receiver
- Parameterized data channel numbers, from 1 to 15
- Parameterized serializer/deserializer factors, support x2, x4, x7, x8
- Either external PLL or internal PLL
- Only support the DDR(double data rate) mode

The LVDS transmitter IP block diagram is shown as below:

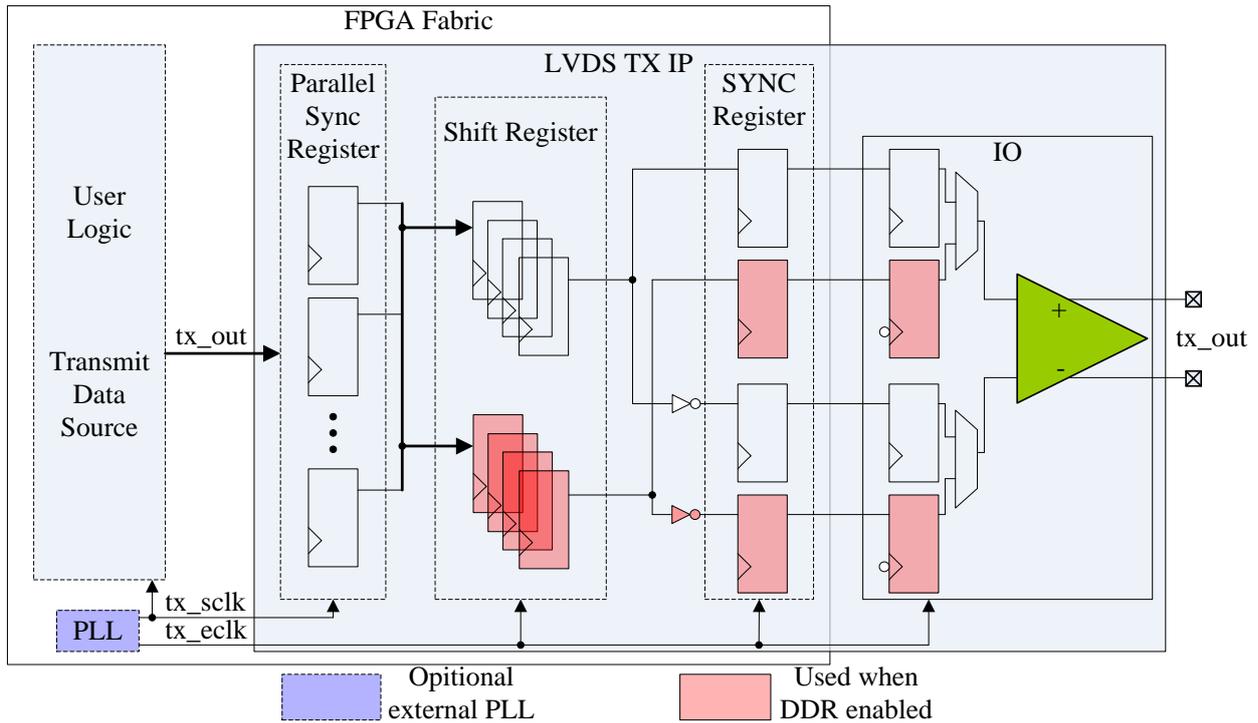


Figure 4 LVDS transmitter IP block diagram

Each LVDS TX channel consists of 3 sets of registers. The first set of registers captures/synchronizes the parallel data from the user Logic using the LVDS tx_sclk. After the data is captured, the data is then loaded into a shift register, which shift the LSB first and the MSB last, two bits per tx_eclk cycle when DDR is enabled. Then the LSB of the shift register feeds into a sync register, and then the shifted serial data is transferred from the sync register into LVDS IO pad using the tx_eclk. Hence, LSB bits precede MSB bits in the output bitstream.

LVDS receiver IP block diagram is shown as below:

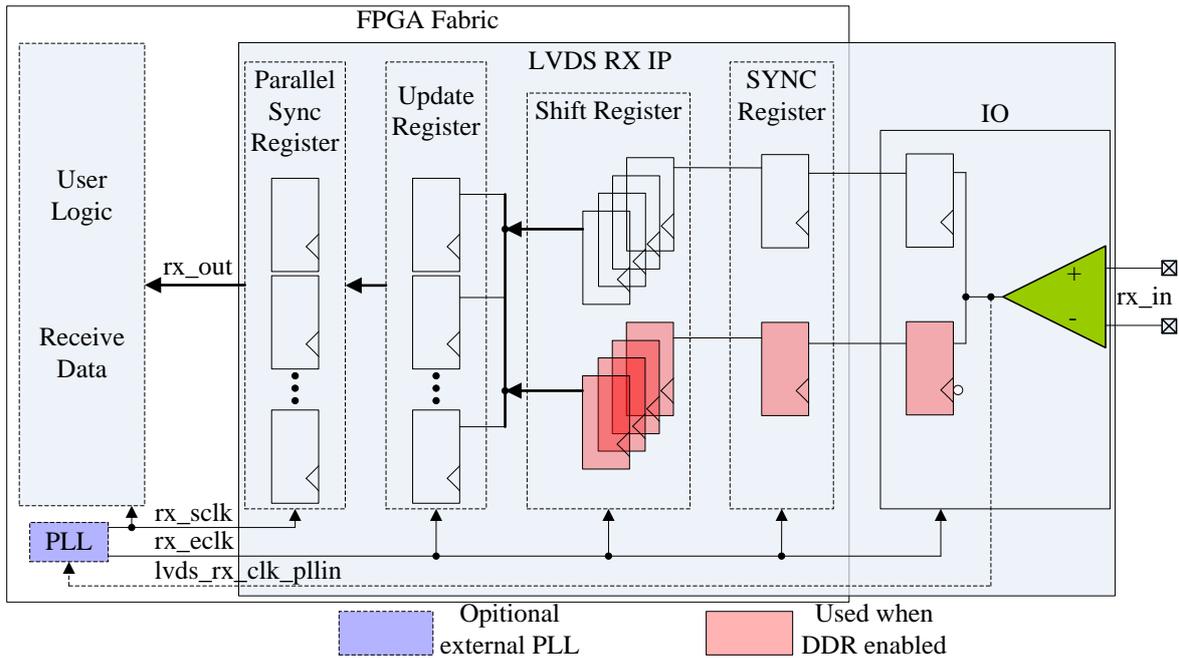


Figure 5 LVDS receiver IP block diagram

Each LVDS RX channel consists of 4 sets of registers. The first set of registers synchronizes the data from LVDS IO pad using the LVDS tx_eclk. Then the data is shifted into a shift register, two bits per tx_eclk cycle when DDR is enabled. And the data stored in the shift register will be aligned into the update register using the LVDS tx_eclk, with first shifted bit being LSB and last shifted bit being MSB in parallel data. At last the update register feeds into a parallel sync register using the LVDS tx_sclk.

In the LVDS IP TX module, all the channels including clock and data channel are the same; and in the LVDS IP RX module, each data channel is the same, while the clock channel will drive a "lvds_rx_clk_pll" pin out to PLL, and align the received data to the received clock.

User can add LVDS IP into design through Fuxi IP Wizard

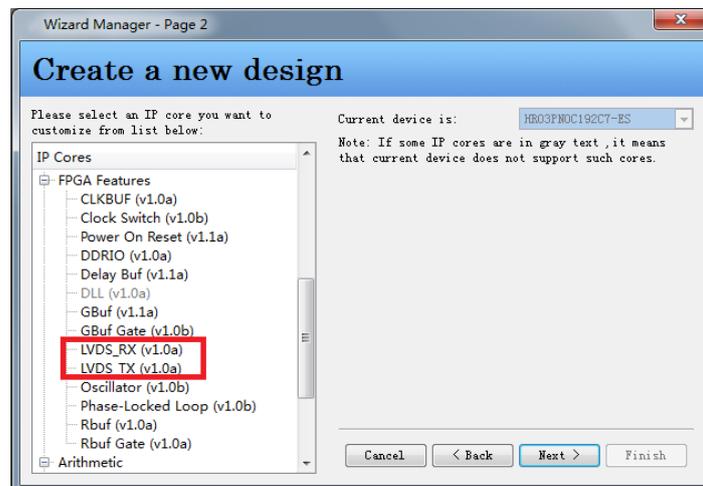


Figure 6 Select LVDS IP from IP list in IP Wizard

In the LVDS IP wizard, user can set the LVDS channel number, serialization and deserialization factor. Furthermore, user can choose to let IP wizard help to add PLL to the LVDS IP (If the PLL is special for the LVDS IP) or add PLL manually (For PLL resource consideration, this PLL might be shared with other logic).

For more information about the LVDS soft IP, please refer to: *HME_LVDS_user_guide_EN01.pdf*

3.4 LVDS IO assignment in Fuxi IO Editor

User can open the Fuxi IO Editor after synthesis and mapping. (Please make sure you have run the synthesis and mapping before you open the IO Editor).

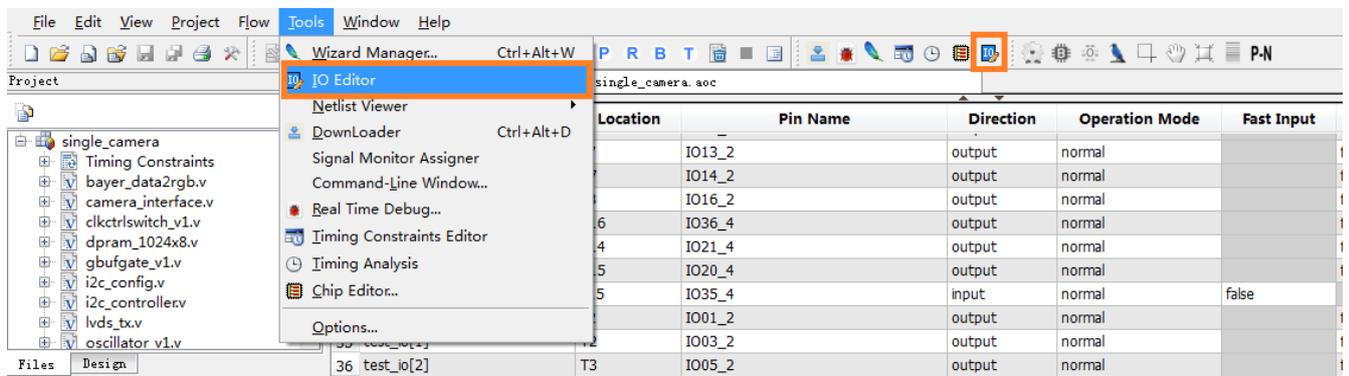


Figure 7 Open IO Editor in Fuxi

User can open the IO Editor from “tools” list or directly using the shortcut key:

For the LVDS IO, the LVDS pair will be list together and the operation mode will be automatically changed to “LVDS”, as shown below:

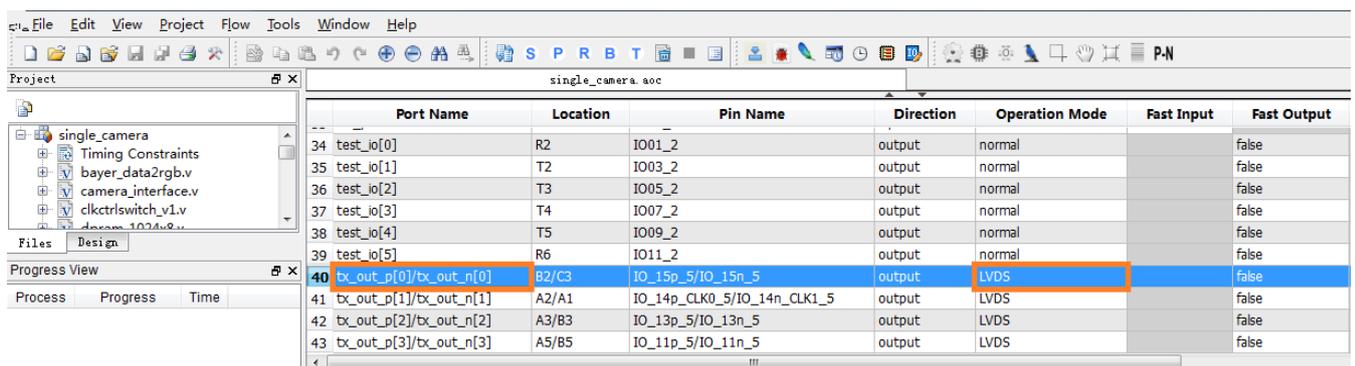


Figure 8 LVDS IO in Fuxi IO Editor

To assign the LVDS location, user can edit the “Location” of LVDS IO, to make it easy for user, the IO Editor will automatically filter the “non-lvds” IO and only keep the available LVDS IO location for user to select, user can assign the LVDS IO location through the “Location” or “Pin Name”. The “Location” and the “Pin Name” are one to one correspondence, when “Location” (“Pin Name”) change, the “Pin

Name (“Location”) will change accordingly.

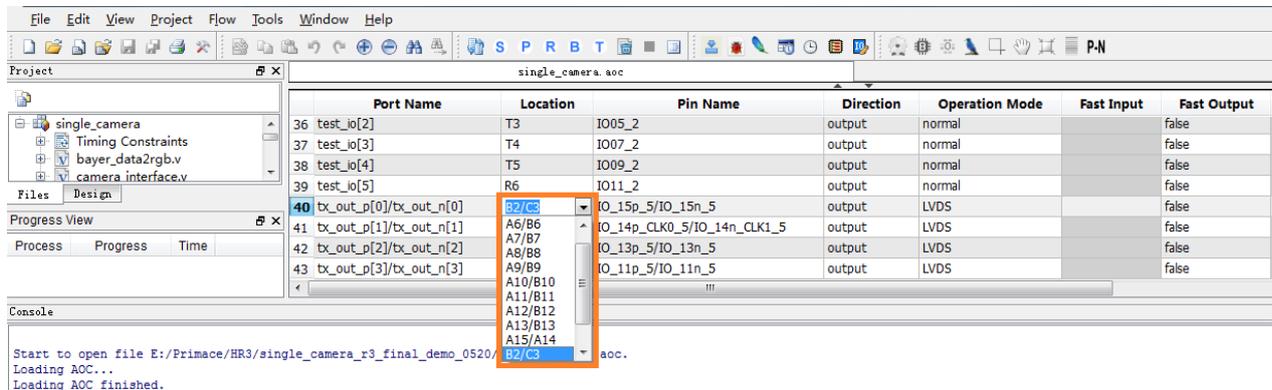


Figure 9 Assign LVDS IO through “Location” in IO Editor

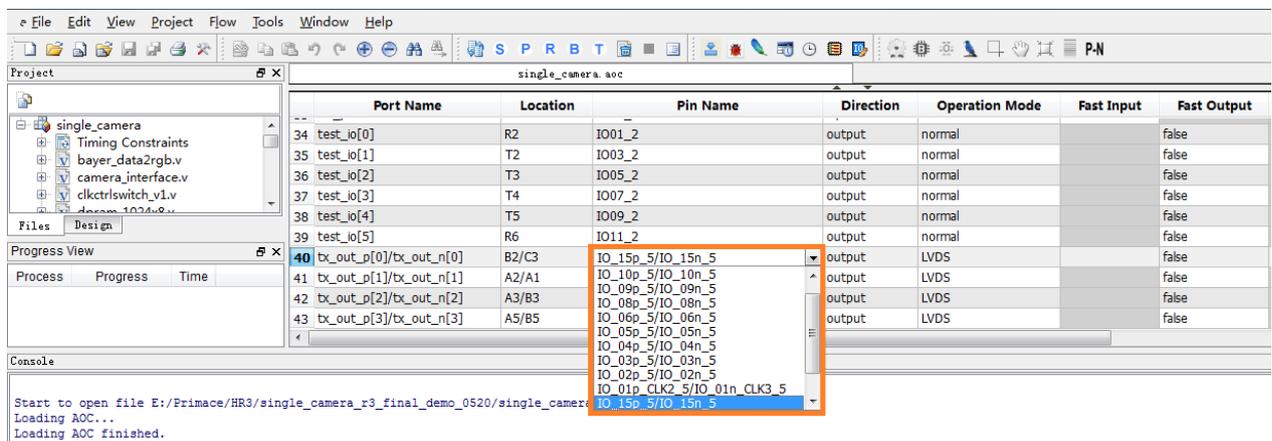


Figure 10 Assign LVDS IO through “Pin Name” in IO Editor

For the LVDS IO, does not support assign LVDS “P” and LVDS “N” IO separately, the IO location is assigned in pair.

3.5 Add timing constraints for LVDS IP and verify

After synthesis & mapping and IO assignment, user need to add timing constraints for LVDS IP. Add timing constraints and timing verification is a very import step in the LVDS interface design. Under different LVDS channel number and serialization (deserialization) factor requirements, the timing requirements on user logic frequency and LVDS Tx/Rx logic clock frequency are different. User need to check if the Fuxi results can satisfy the timing requirements especially when high clock frequency is required.

Before adding timing constraints, user need to run synthesis and mapping first. User can add timing constraints through Fuxi Timing Constraints Editor.

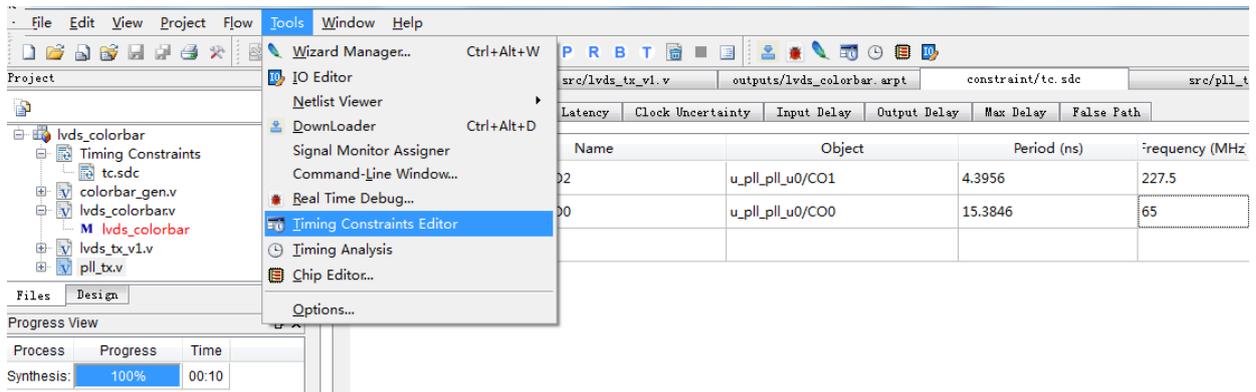


Figure 11 Add timing constraints through Timing Constraints Editor

After lunch the Timing Constraints Editor, user need to constraint the clock frequency of user logic and LVDS Tx/Rx logic. The following diagram shows an example of adding timing constraints.

Enable	Name	Object	Period (ns)	Frequency (MHz)	Comment
<input checked="" type="checkbox"/>	u_pll_pll_u0/CO1	u_pll_pll_u0/CO1	4.3956	227.5	LVDS Tx Loigc Frequency
<input checked="" type="checkbox"/>	u_pll_pll_u0/CO0	u_pll_pll_u0/CO0	15.3846	65	User Logic Frequency
<input checked="" type="checkbox"/>					

Figure 12 Timing constraints example

In this example, the user logic is 65MHz and the serialization factor is x7, so the LVDS Tx logic frequency is 227.5MHz, set the timing constraints and add to the project.

After add timing constraints, run through P&R (Placer and router) and then check the timing report and verify if all timing constraints are satisfied. User can get the timing report through: Timing Analysis

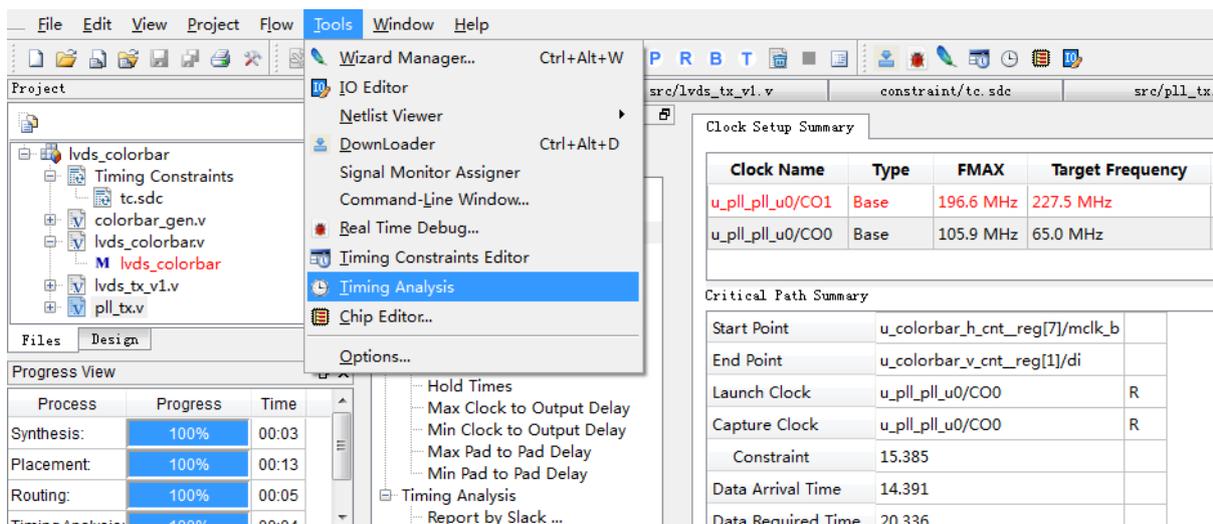


Figure 13 Get timing report through Timing Analysis

The flowing timing report is an example report with timing violation. The target frequency is 227.5MHz, but the timing report is only 196.6MHz.

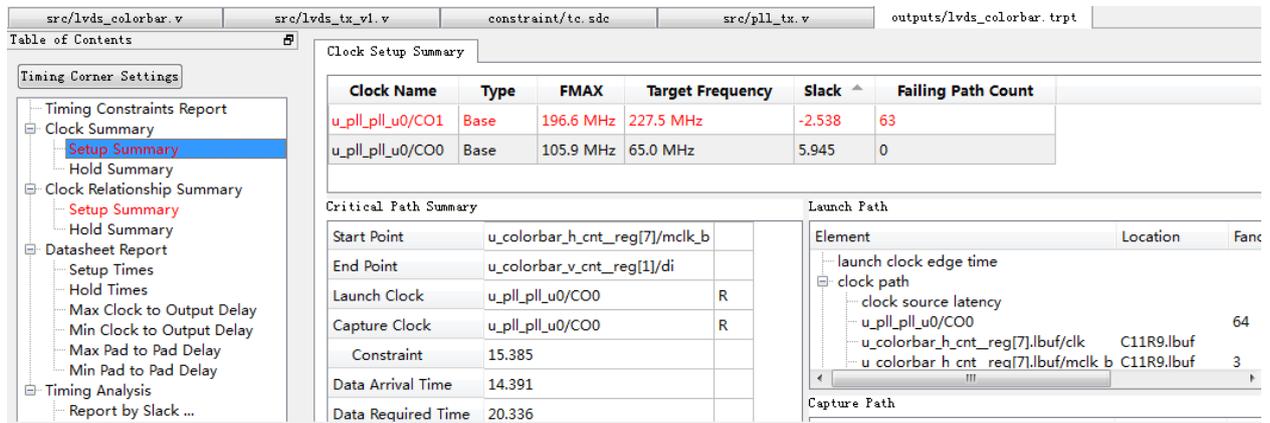


Figure 14 Example of timing report

If there are timing violations, two ways can be considered:

- Adjust timing constraints, add more strict timing constraints, like change the target frequency from 227.5MHz to 250MHz or add release timing constraints, like change the target frequency from 227.5MHz to 200MHz. After adjust timing constraints, rerun placer, router and check timing again. This need to have a try to check if can get better timing results. Fuxi also provide a tool: iXplore, which helps users achieves the best performance by automatically scanning various options and timing constraints. For detailed information about the iXplore, please refer to the user guide of Fuxi: [Fuxi_User_Guide_En.pdf](#).
- Check the failed path and get the critical path information to see if RTL can be further optimized.

3.6 Board design considerations on LVDS IO

The LVDS I/O standard is a high-speed, low-voltage swing, low power, and general purpose I/O interface standard. The HR FPGA device meets the ANSI/TIA/EIA-644 standard.

LVDS and SubLVDS inputs require external compensation and termination resistors for proper operation, as shown in the figure below. A termination resistor, RT, between the positive and negative inputs at the receiver forms a current loop. The current across this resistor generates the voltage detected by the receiver’s differential input comparator.

Similarly, HR FPGA LVDS and SubVLDS outputs require an external resistor network, consisting of two series resistors, RS, and a parallel resistor, RP. This resistor network adjusts the FPGA’s output driver to provide the necessary current and voltage characteristic s required by the specification.

The signals are routed with matched trace impedance on the printed circuit board, typically with 50Ω impedance.

The figure below shows a typical LVDS I/O channel with the compensation resistors connected. To guarantee the HR FPGA LVDS can work well on-board, the proper compensation resistors must be connected on board.

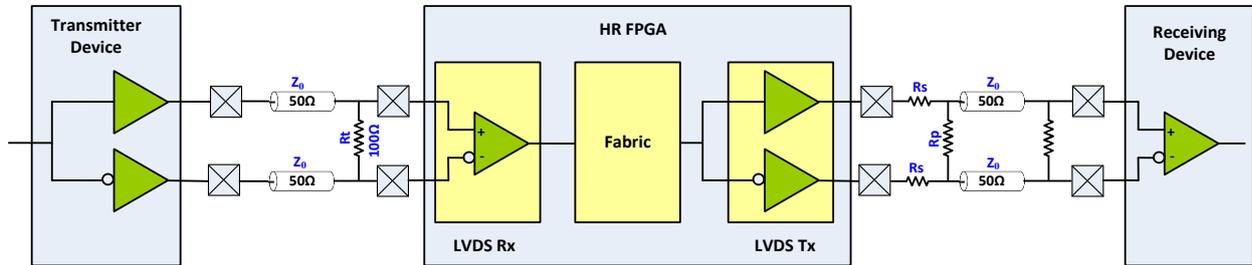


Figure 15 Typical LVDS differential I/O channel

Table 4 Compensation Resistor

Variable	Description
Z_0	Characteristic impedance of the printed circuit board trace; data sheet values assume 50Ω traces
R_t	LVDS/SubLVDS input termination resistor
R_p	LVDS/SubLVDS output source compensation parallel resistor
R_s	LVDS/SubLVDS output source compensation series resistor

Table 5 Typical resistors using in LVDS

Variable	LVDS (2.5v) (ohm)	SubLVDS (1.8v) (ohm)
R_s	90.83	163.3
R_p	120	140
R_t	100	100

3.7 On-board functionality verification and debug

After timing check, user can generate the bitstream and download on board. If on-board functionality verification fails, Fuxi provides two ways to help user debug the design: Debugware and Signal monitor.

The Debugware is an embedded logic analyzer. Designer always wants to inspect the signal transition inside an FPGA. The Debugware uses the embedded memory to store the internal signals waveform, and retrieves data via JTAG port after triggering events happened.

Signal monitor is a tool that allows user to route user-specified signals to output pins without affecting the existing design, so that you can debug signals without recompiling the design. Incremental routing method is applied in this flow.

For detailed information about Debugware and Signal monitor, please refer to the user guide of Fuxi:

[Fuxi_User_Guide_En.pdf](#)

4 Typical LVDS Application Example

This chapter describes 3 typical LVDS application examples: LVDS loop back example, LVDS 7:1 video interface example and digital CDR SerDes example based on LVDS interface, which help user to quickly familiar how to design LVDS related application based on HR FPGA device.

4.1 LVDS loop back example

The loopback example described in this chapter uses the HR FPGA to implement both the 7:1 transmitter and receiver. The figure below shows the design implementation. 7-bit transmit data is generated in the FPGA and then serialized and transmitted as four bits of LVDS data using the 7:1 transmitter logic. The 1-bit LVDS data is then looped back into the FPGA receiver side and deserialized using the 7:1 receiver logic. This deserialized data is then fed to the data compare logic module which compares the deserialized receiver data to the original counter values transmitted. The compared results show on the LED.

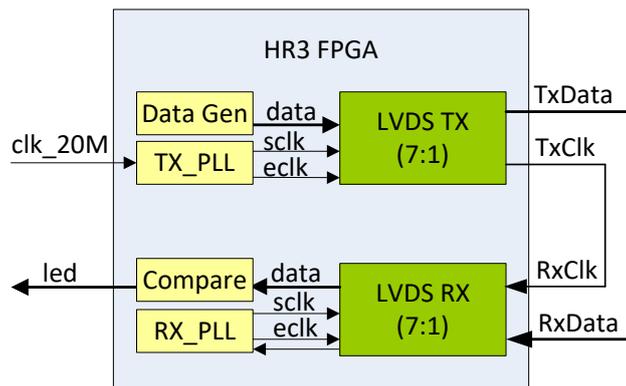


Figure 16 LVDS loop back example

In this example, with TX/RX sclk clock 60MHz, edge clock 210MHz, serial/deserial-factor 7, 1 clock channel and 1 data channel.

This case consists of 6 parts as shown in above figure: TX_PLL, LVDS TX, LVDS RX, RX_PLL, Data Generator, Data comparator

- TX_PLL

This block generates the 60MHz sclk and the 210MHz eclk for LVDS TX block. Its reference input clock is a 20MHz Oscillator.

- LVDS TX

This block serializes the input parallel const data into LVDS signals, with sclk 60MHz, eclk 210MHz, serial-factor 7, 1clock channel and 1 data channel.

■ LVDS RX

This block de-serializes the input LVDS signals, which is coming from its self's LVDS TX, into parallel data, which should be a const data, and this data is connect to led. And this block has sclk 60MHz, eclk 210MHz, deserial-factor 7, 1clock channel and 1 data channel.

■ RX_PLL

This block generates the 60MHz sclk and the 210MHz eclk for the LVDS RX block. Its reference input clock is coming from the LVDS RX block's clock channel, which is 60MHz.

■ Data generator

The data generator generates the data for LVDS transmitter.

■ Data comparator

The data comparator compares the received data from LVDS receiver and compare with the expected data and show the results on LED

Based on this example, user can easily build its own LVDS transmitter or receiver system. When user add LVDS IP into design through IP wizard, this example will be provided for user as reference.

4.2 LVDS 7:1 video interface example

Source synchronous interfaces consisting of multiple data bits and clocks have become a common method for moving image data within electronic systems. A prevalent standard is the 7:1 LVDS interface (employed in Channel Link, Flat Link, and Camera Link), which has become a common standard in many electronic products including consumer devices, industrial control, medical, and automotive control.

Using LVDS interface to drive the LCD, the data and clock are received or transmitted to or from the FPGA in LVDS format, with the data at relatively high speed. The exact speed depends on the resolution, frame rate and color depth used by the display. For example, 800x600 to 1024x768 displays require LVDS data to be transmitted from 40 MHz to 78.5 MHz for 60 Hz to 75 Hz refresh rates. This translates to LVDS data rates of 280 Mbps to 549 Mbps. Higher resolution displays, such as 1280x1024 60Hz, require data to be transmitted with 108 MHz LVDS clocks. For this system, data will transmit at 756 Mbps.

openLDI (open LVDS Display Interface) and FPDI (Flat Panel Display Interface) are two usual display interface standard. Both standards have common specification on RGB bit mapping in a pixel clock period, as shown in following figures.

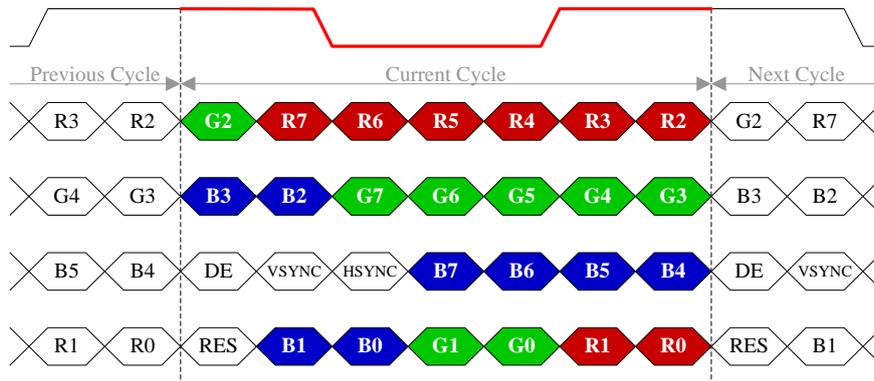


Figure 17 LVDS 7:1 LDI data format

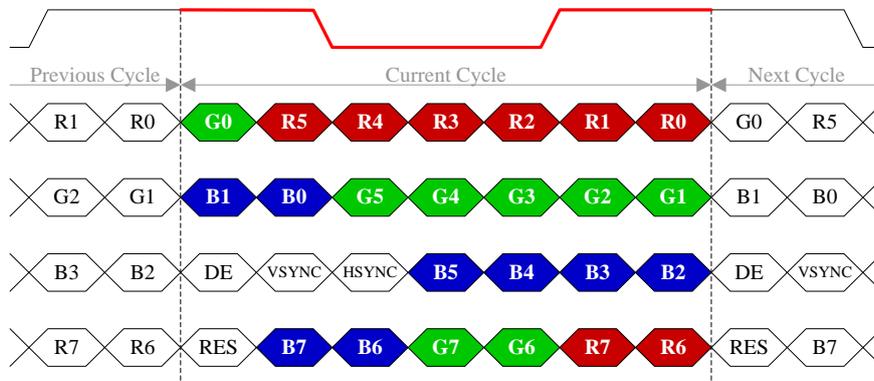


Figure 18 LVDS 7:1 FPD data format

In this example, 7:1 LVDS interface is used to drive the LCD display, with 1024x768 resolution, 60Hz, the LVDS data rates is 455Mbps and the LVDS 7:1 FPD data format is used.

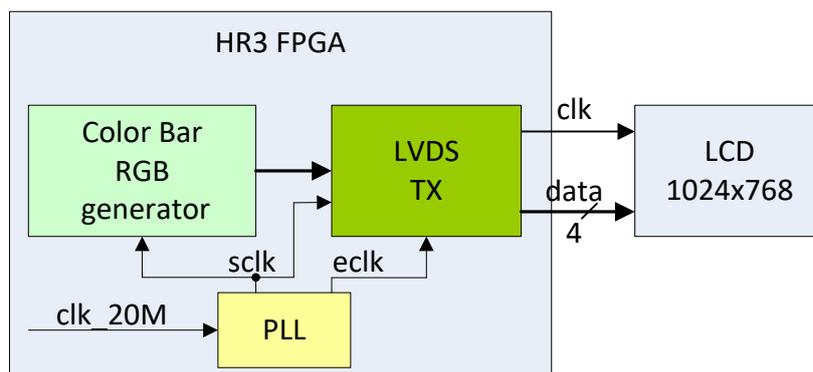


Figure 19 LVDS 7:1 video interface example

This case consists of 3 parts as shown in above figure: Color Bar RGB&DE generator, LVDS TX, and PLL.

- Color Bar RGB generator:

This block generates the color bar RGB and DE signals, then push these data into LVDS TX parallel data inputs.

■ LVDS TX:

This block serializes the input parallel data into LVDS signals, driving the LVDS LCD, the LVDS TX contains 1 clock channel and 4 data channels, with TX sclk or pixel clock 65MHz, edge clock 227.5MHz, serial-factor is 7.

■ PLL:

This block generates the 65MHz sclk, and the 227.5MHz eclk (**eclk must be 3.5*sclk**), with 227.5MHz output clock delay 90 degree.

Based on this example, user can easily build their video or image display system. When user add LVDS IP into design through IP wizard, this example will be provided for user as reference.

The following diagram shows a typical application based HR LVDS interface design.

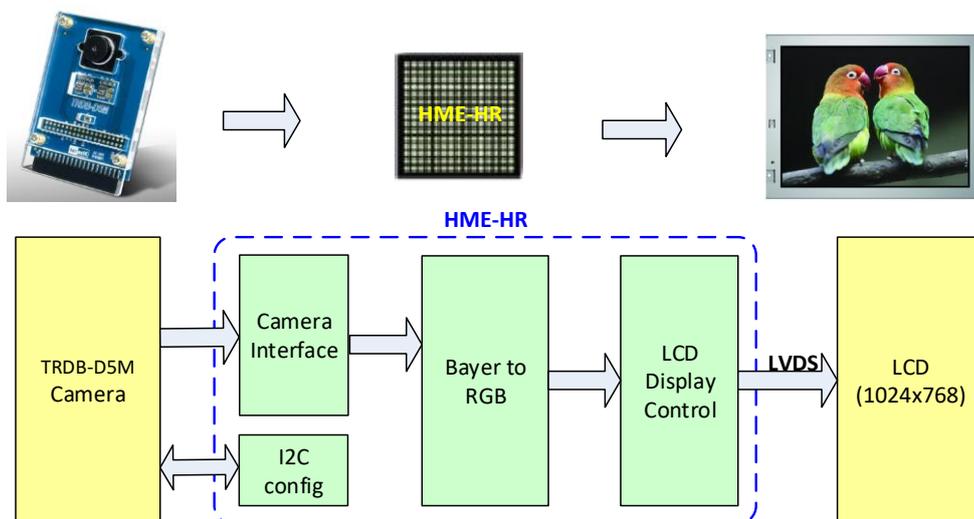


Figure 20 Typical application based on HR FPGA LVDS interface

In this application, HR FPGA provides the driver and config data for the TRDB-D5M camera and gets the raw data from camera. The camera data is transformed from bayer format to RGB format and drive the LCD through LVDS interface.

For more information of this application demo, please contact with us support@hercules-micro.com

4.3 Digital CDR SerDes interface example

SerDes is the called for short of “Serializer” and “Deserializer”, which is a very common serial communication technic. This chapter describes an example of using HR FPGA LVDS interface to implement SerDes function.

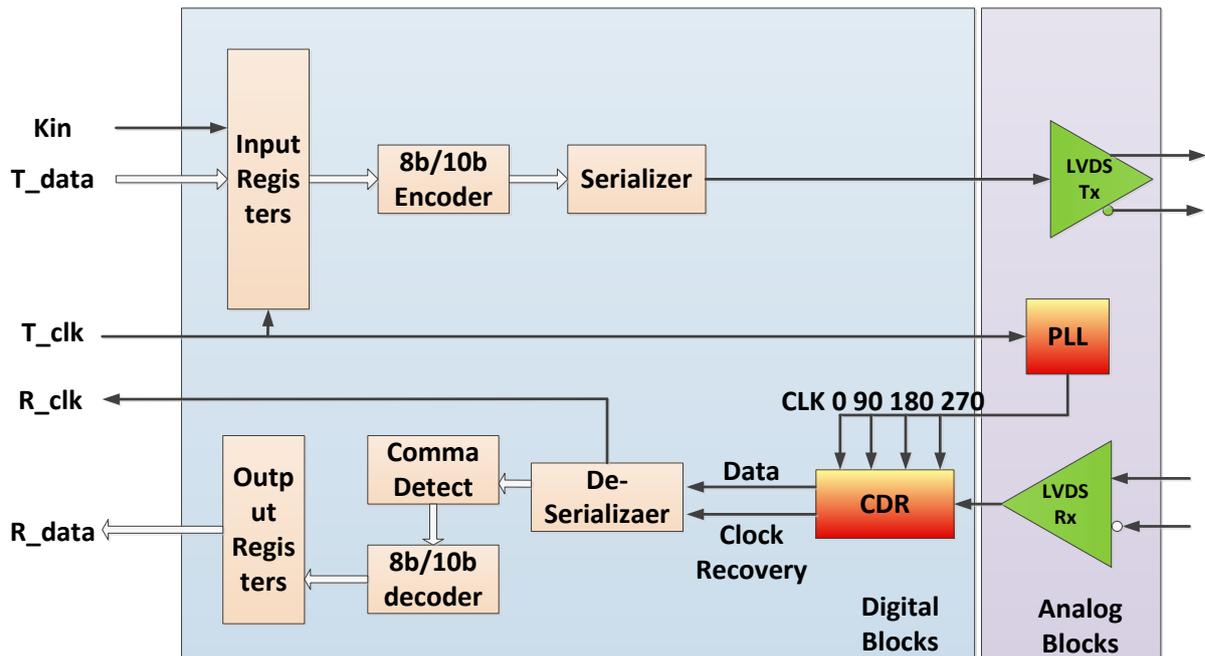


Figure 21 Block diagram of SerDes based on LVDS interface

The SerDes function mainly includes 2 parts: transmission block and reception block.

Transmission block includes 8b/10b encoder, serializer, PLL and LVDS transmitter. The transmission block gets the data from user logic and encodes the data with 8b/10b encoder, then the encoded data are serialized into serial data to LVDS transmitter. The LVDS transmitter will transform the data from CMOS voltage to differential signal to improve the anti-noise ability.

Reception block includes 8b/10b decoder, comma detector, deserializer, CDR (Clock Data Recovery) and LVDS receiver. The reception block transforms the differential input signal into CMOS voltage signal through LVDS receiver. The deserializer transforms the serial data into 10 bits parallel data. The boundary of the data can be detected through comma detector. The data are then transformed to 8 bits data through 8b/10b decoder and transmit to user logic.

The CDR (Clock Data Recovery) is the key of the SerDes. The function CDR is shown as below:

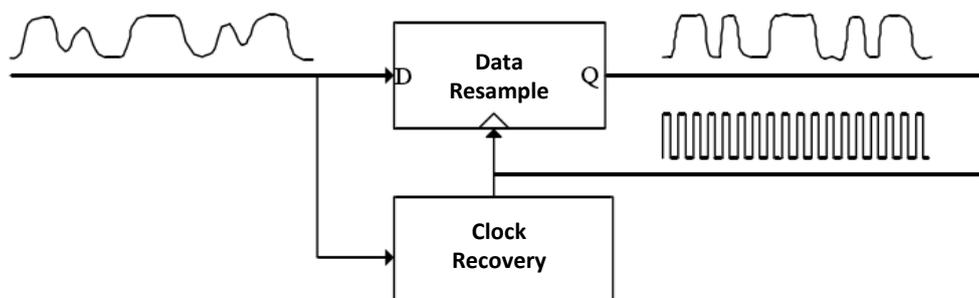


Figure 22 CDR function

The function of CDR includes:

- Recover the clock from the data stream (usually the NRZ encode data stream)
- Using the recovered clock to resample the data

In this example, 4 phase oversampling method is used to implement the CDR function.

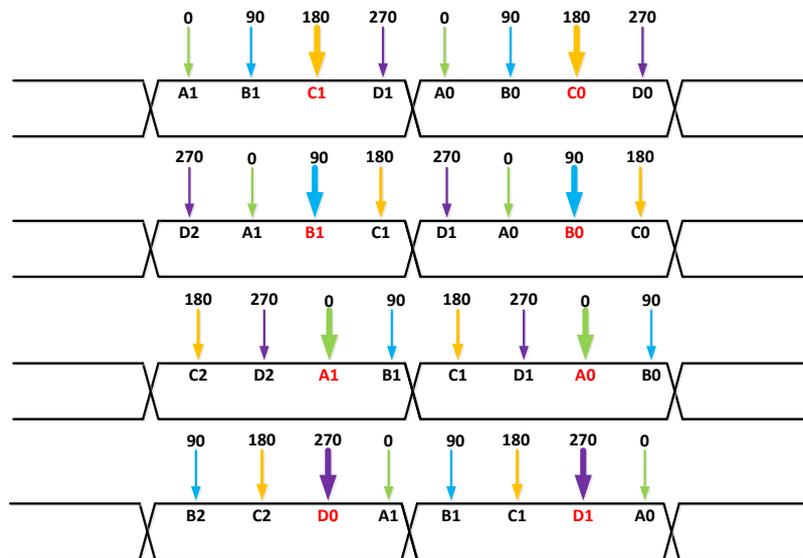


Figure 23 CDR implement principle

Figure above shows the basic principle of CDR implementation. Using 4 phases of clock (0, 90, 180, 270) to sample the clock input and find out the best sample location to get the final recovered clock.

Based on HR3 FPGA, this example can get to 270Mbps performance.

This example can be easily extended to support: STM-1 155Mbps SDH interface, Standard definition SDI interface or user defined serial communication interface.

For more information about this example, please contact with us: support@hercules-micro.com

5 Revision History

Date	Doc Version	Comments
Aug. 2014	V1.0	Initial release
Oct. 2018	V2.0	Update Figure6~Figure11、 Figure13