XILINX®

XAPP245 (v1.1) March 15, 2001

Eight Channel, One Clock, One Frame LVDS Transmitter/Receiver

Author: Ed McGettigan

Summary

This application note describes a 5.12 Gbps transmitter and receiver interface using ten Low-Voltage Differential Signalling (LVDS) pairs (one clock, eight data channels, one frame) implemented in a Virtex™-E FPGA. The accompanying library of designs targets Virtex-E devices. The design is implemented as a EDIF netlist with embedded location constraints and VHDL and Verilog simulation files. The design does not rely on guide files for successful performance.

Introduction

In Figure 1 the LVDS system design example includes a transmitter on the left of the diagram and a receiver on the right. The transmitter implements 8 to 1 serialization. This is why the DATA input port of the transmitter is 64-bits wide, while the TXN_data / TXP_data differential pair lines are only 8-bits wide. The TXP_clk and TXN_clk differential pair transmits the source-synchronous clock to the receiver. The TXP_fr and TXN_fr differential pair transmits the FRAME signal to the receiver.

On the receiver exists a set of signal ports complementary to those on the transmitter. Data, clock, and frame lines are received on the receiver input ports with the clock delayed by 700 ps with respect to the data and the frame signals. On the output of the receiver are aligned versions of the 64-bit received data and frame in the SYSCLK clock domain. Read enable (RE), RESET, and SYSCLK are user controlled signals. BUFSTAT reports the status of the receive buffer.

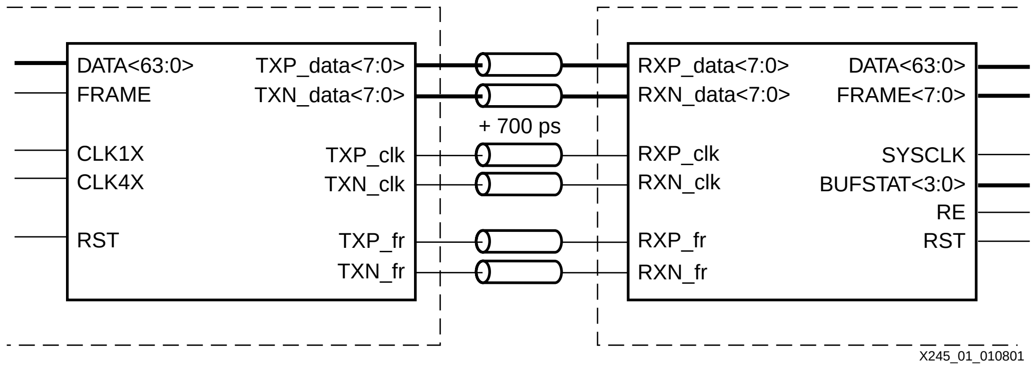


Figure 1: **LVDS System Design Example**

Transmitter

Figure 2 is the transmitter module.The module pin descriptions are listed in Table 1.

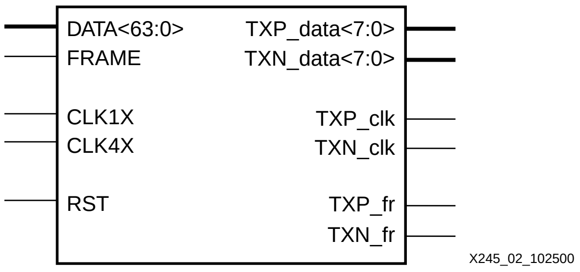


Figure 2: **Transmitter**

Table 1: **Transmitter Module Pin Definitions**

I/O Type	Module Pin Name	Definition
Input	DATA <63:0>	Data presented at the 64-bit data input bus is transmitted to the receiver in LSW to MSW order. E.g. <7:0>, <15:8>, <23:16>, <31:24> . . . <63:56>
	Frame	Indicates the start of the frame and is useful in the synchronization of the transmitter and receiver. Because of the internal construction of the transmitter, no frame may be smaller than 64 bits. There is no maximum frame size.
	CLK1X	System clock
	CLK4X	LVDS transmit clock
	RST	Reset
Output	TXP_data<7:0> TXN_data<7:0>	The signals that comprise the differential pairs which form the 8-bit transmit bus. All data transferred over this bus is synchronous with the transmit clock.
	TXP_fr, TXN_fr	TXP_fr and TXN_fr form the differential Frame signal.
	TXP_clk, TXN_clk	TXP_clk and TXN_clk form the differential, source-synchronous clock signal.

The transmitter block diagram is shown in Figure 3.

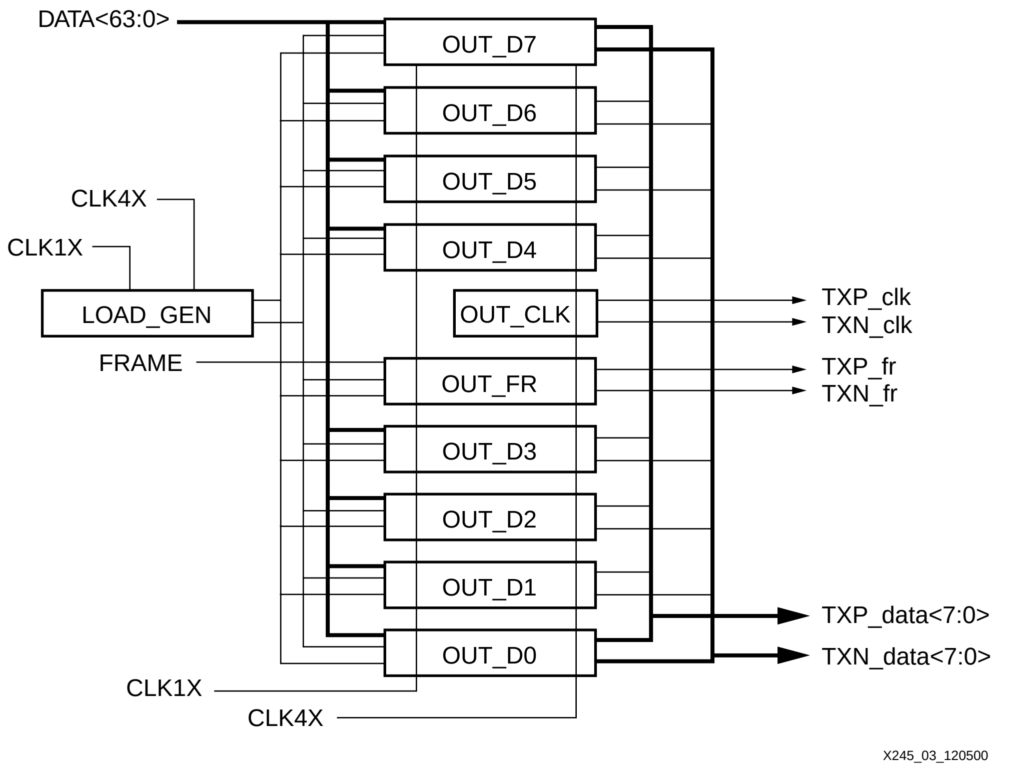


Figure 3: **Transmitter Block Diagram**

The transmitter is composed of three major modules. Each of these is discussed in detail in the following sections. Moving from left to right, first is the LOAD_GEN module. Above and below LOAD_GEN, in a vertical column, are the DDR output stages of the transmitter OUT_D[x]. Nestled in the middle of the transmitter is the OUT_CLK module, which transmits the source synchronous transmit clock.

Figure 4 has the transmitter output waveforms. All outputs of any given byte leave the device at the same time. TXP_fr is held High for four clock edges and stays Low until the next frame assertion.

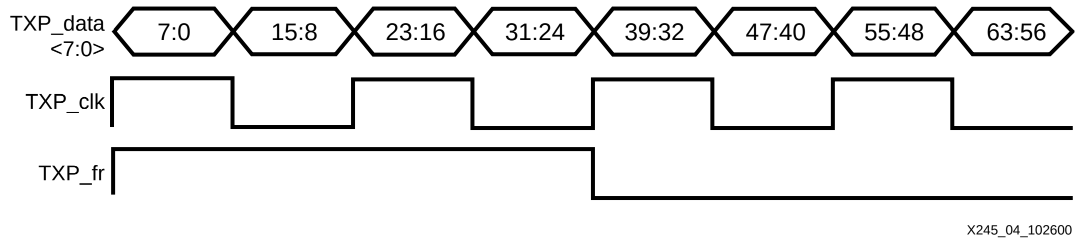


Figure 4: **Transmitter Output Waveforms**

LOAD_GEN Overview

Figure 5 shows the LOAD_GEN module. The LOAD_GEN module generates the load pulses for the parallel to serial converters in the OUT_D[X] modules. Timing of the load signal is demanding as it drives the individual data channels that span up to 13 rows at 320 MHz (3.125 ns). The LOADRISE and LOADFALL signals are replicated for distribution to the upper and lower OUT_D[X] modules. This minimizes the overall length and loading of each signal. Examining the gate-level implementation, all load signals with a "<0>" suffix drive the data channels above the LOAD_GEN module and "<1>" for channels below the LOAD_GEN module. Figure 6 shows the LOAD_GEN waveforms. The module pin descriptions are listed in Table 2.

Table 2: **LOAD_GEN Module Pin Definitions**

I/O Type	Module Pin Name	Definition
Input	CLK1X	Clock for DATA domain (80 MHz)
	CLK4X	Clock for DDR domain (320 MHz).
Output	LOADRISE <1:0>	Load pulse for rising edge data
	LOADFALL <1:0>	Load pulse for falling edge data

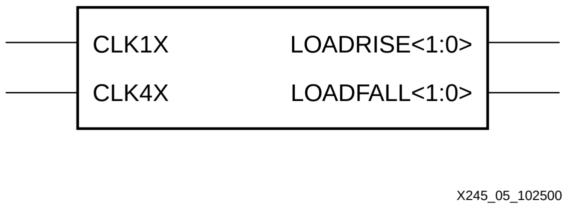


Figure 5: **LOAD_GEN Module**

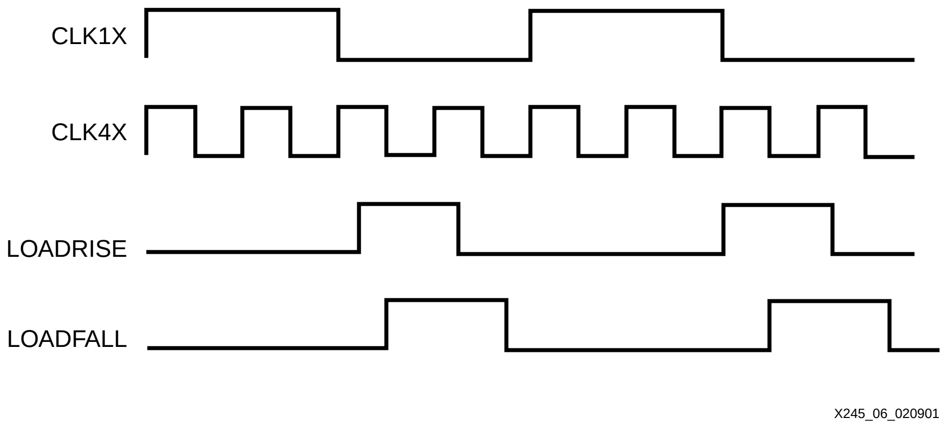


Figure 6: **LOAD_GEN Waveforms**

OUTSTAGE Overview

Figure 7 is the OUTSTAGE module. The OUTSTAGE block performs as an 8-bit serializer. The 8-bit data from the CLK1X domain is loaded in two 4-bit parallel-to-serial converters. The even bits (6, 4, 2, 0) are transferred during the falling edges of CLK4X and a rising edge of TXP_clk. The odd bits (7, 5, 3, 1) are transferred during the rising edges of CLK4X and a falling edge of TXP_clk. Data will exit the OUTSTAGE module from LSB to MSB. Figure 8 shows the OUTSTAGE block diagram. Figure 9 shows the OUTSTAGE waveforms. The module pin descriptions are listed in Table 3.

Table 3: **OUTSTAGE Module Pin Definitions**

I/O Type	Module Pin Name	Definition
Input	DATA <7:0>	Slice of data for this transmitter
	CLK1X	Clock for the DATA bus
	CLK4X	Clock for the DDR logic
	LOADRISE	Load data on the next rising edge of CLK4X
	LOADFALL	Load data on the next falling edge of CLK4X
	RST	Reset
Output	TXP, TXN	LVDS output data

Figure 7

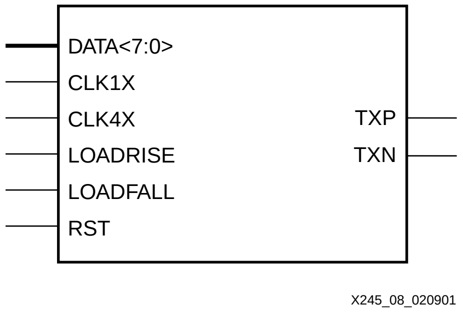


Figure 7: **OUTSTAGE Module**

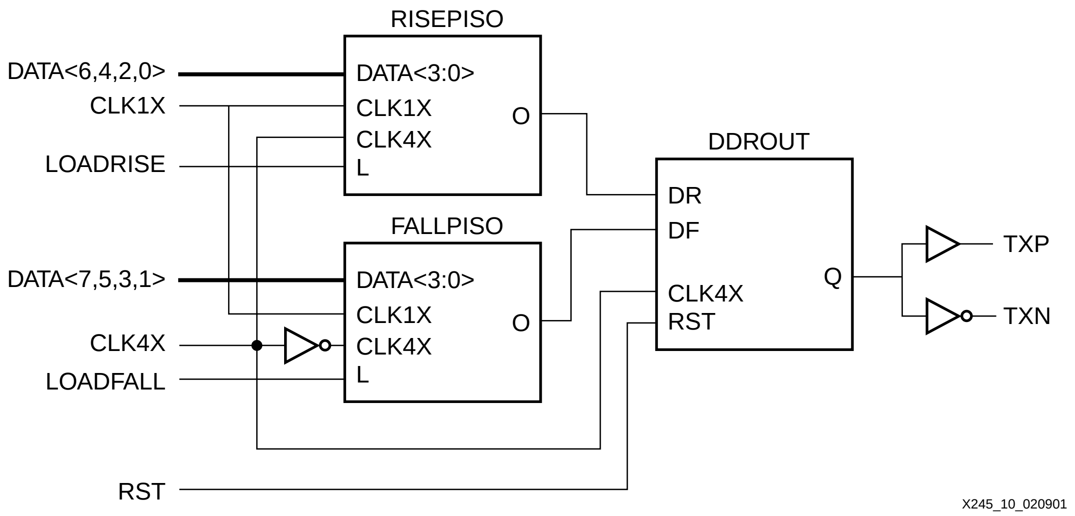


Figure 8: **OUTSTAGE Block Diagram**

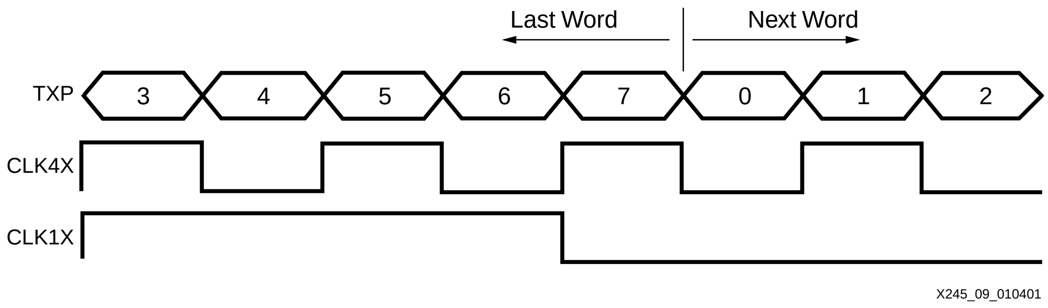


Figure 9: **OUTSTAGE Waveforms**

PISO Overview

Figure 10 is the Parallel-In, Serial-Out (PISO) module. The PISO block is instantiated twice in each outstage. One PISO block is triggered off the rising edge of CLK4X, while the other is triggered off the falling edge. To ease timing constraints, data is registered into the PISO on the CLK1X domain. Data is not transferred from the CLK1X to the CLK4X domain until the third rising edge (or falling edge) of CLK4X, allowing data from the 1X clock domain to have ample settling time. The signal descriptions are shown in Table 4.

Table 4: **PISO Module Pin Definitions**

I/O Type	Module Pin Name	Definition
Input	DATA <3:0>	Parallel data bus
	CLK1X	Clock for the Parallel data
	CLK4X	Clock for the Serial out data
	L	Load data on the next CLK4X edge
Output	O	Serial data output

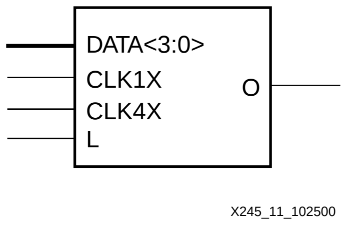


Figure 10: **PISO Module**

DDROUT Overview

Figure 11 is the Double Data Rate Flip-Flop Output (DDROUT) module. The DDROUT block is a critical module in this design and is implemented as a hard macro. Two versions have been created for the left and right sides of the device (DDROUT_left and DDROUT_right). The signal descriptions are shown in Table 5. Timing waveforms are shown in Figure 12.

Table 5: **DDRFD Module Pin Definitions**

I/O Type	Module Pin Name	Definition
Input	DR	Data for the rising edge
	DF	Data for the falling edge
	CLK4X	DDR Clock
	RST	Asynchronous reset
Output	Q	DDR Output

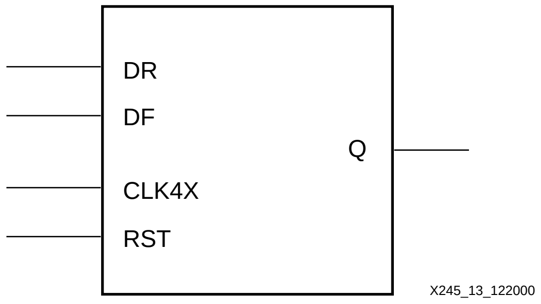


Figure 11: **DDROUT Module**

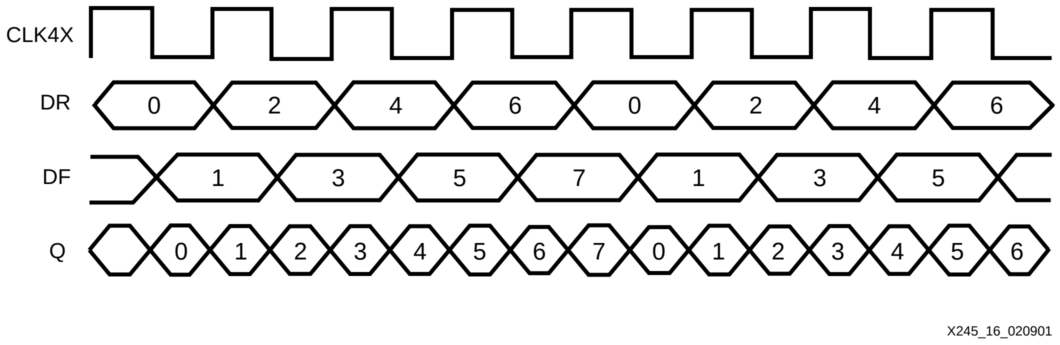


Figure 12: **DDRFD Waveforms**

Transmitter Floorplan

The transmitter implemented in the design library does not require the use of the floorplanner or guide files. Each data channel is implemented as a relatively placed macro (RPM) is absolutely placed for the specific part, package, and location.

The transmitter is limited to 13 rows; six rows above LOAD_GEN and six rows below LOAD_GEN. The clock and frame are centered. The data channels are split evenly on either side. The data channels have LVDS output buffers for the left side that are asynchronous capable in the same row. Figure 13 shows a typical floorplan.

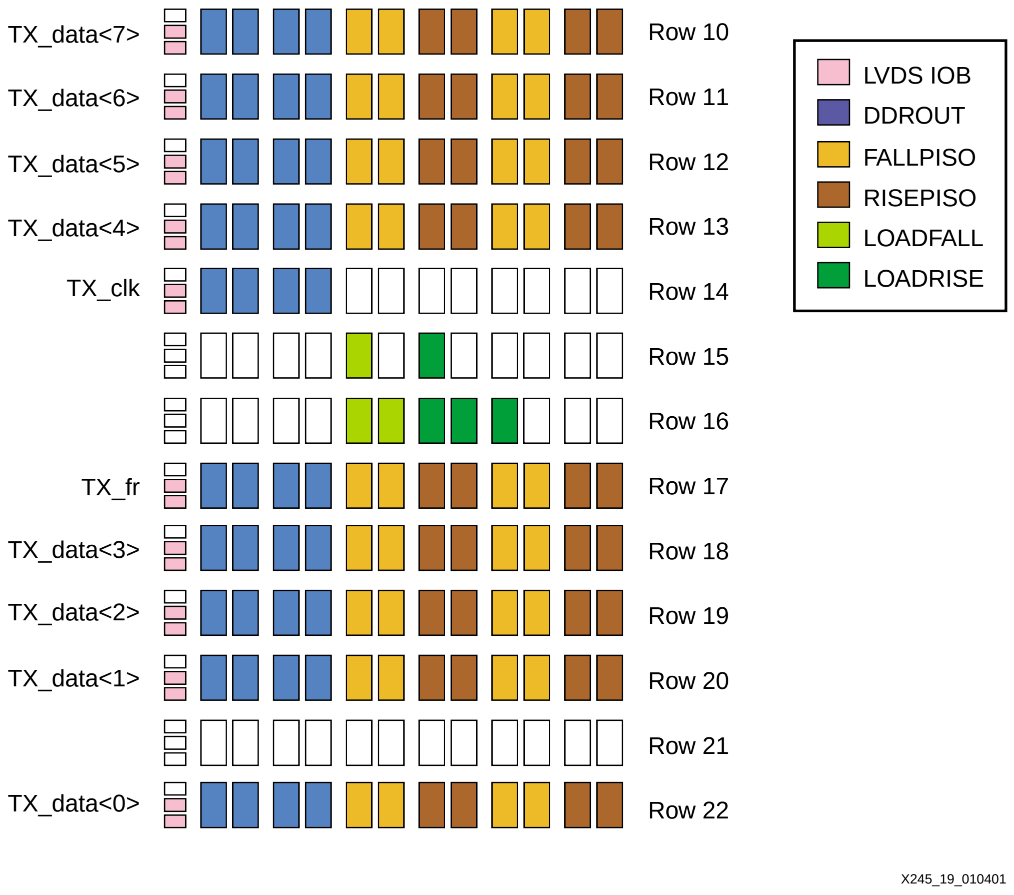


Figure 13: **Transmitter Floorplan**

Timing Issues

CLK4X and CLK1X must be in phase and generated by the same CLKDLL.

Receiver

Figure 14 is the receiver diagram. The signal descriptions are shown in Table 6.The receiver block diagram is shown in Figure 15.

The receiver is composed of three major subcircuits. Each of these is discussed in detail in the following sections. Moving from left to right, first is the HSRX subcircuit. The HSRX is the high-speed LVDS receiver. It passes unaligned data to the FIFO block for queueing to the system clock domain. The FIFO in turn passes the data to the ALIGN block which aligns the incoming data on 64-bit boundaries.

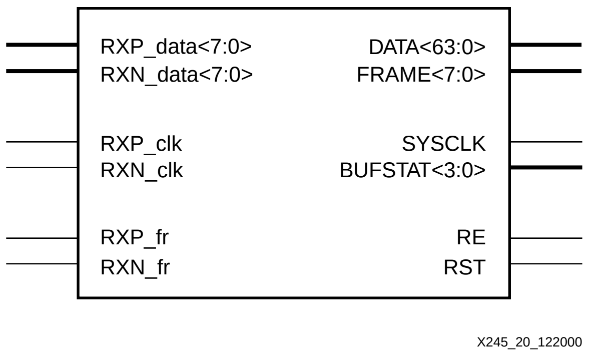


Figure 14: **Receiver**

Table 6: **Receiver Module Pin Definitions**

I/O Type	Module Pin Name	Definition
Input	RXP_data <7:0> RXN_data <7:0>	The signals that comprise the differential pairs which form the 8-bit transmit bus.
	RXP_fr, RXN_fr	Start of Frame flag LVDS input pins.
	RXP_clk, RXN_clk	Receiver module clock.
	SYSCLK	System clock for read FIFO
	RE	FIFO read enable
	RST	Receiver reset
Output	DATA <63:0>	64-bit bus of received data out of FIFO. Synchronous to SYSCLK
	FRAME <7:0>	Start of frame flag (`00001111`)
	BUFSTAT <3:0>	FIFO buffer status

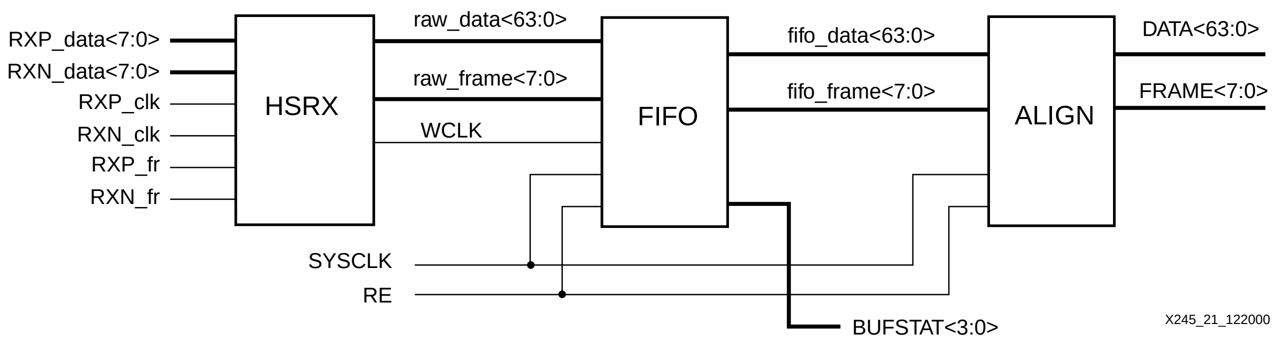


Figure 15: **Receiver Block Diagram**

HSRX Overview

Figure 16 shows the High Speed Receiver module (HSRX). CLKGEN module generates half and quarter cycle phase clocks from the original receive clock. Each channel is an 8 to 1 deserializer. Outputs from each channel are not aligned in HSRX, but in the ALIGN module. Figure 17 shows the HSRX block diagram. The signal descriptions are shown in Table 7.

Table 7: **HSRX Module Pin Definitions**

I/O Type	Module Pin Name	Definition
Input	RXP_data <7:0> RXN_data <7:0>	LVDS data pins
	RXP_clk RXN_clk	LVDS clock
	RXP_fr RXN_fr	LVDS frame
Output	DATA <63:0>	Unaligned data bits
	FRAME <7:0>	Unaligned frame bits
	WCLK	FIFO write clock derived from RXP_clk

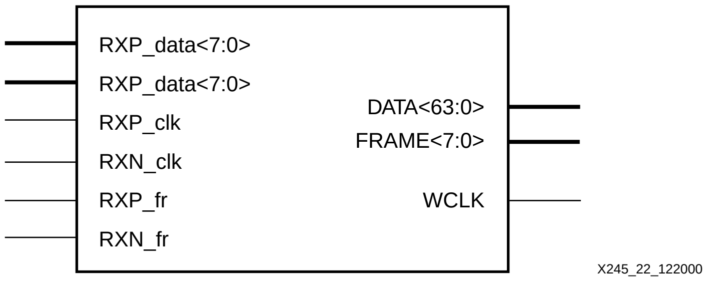


Figure 16: **HSRX Module**

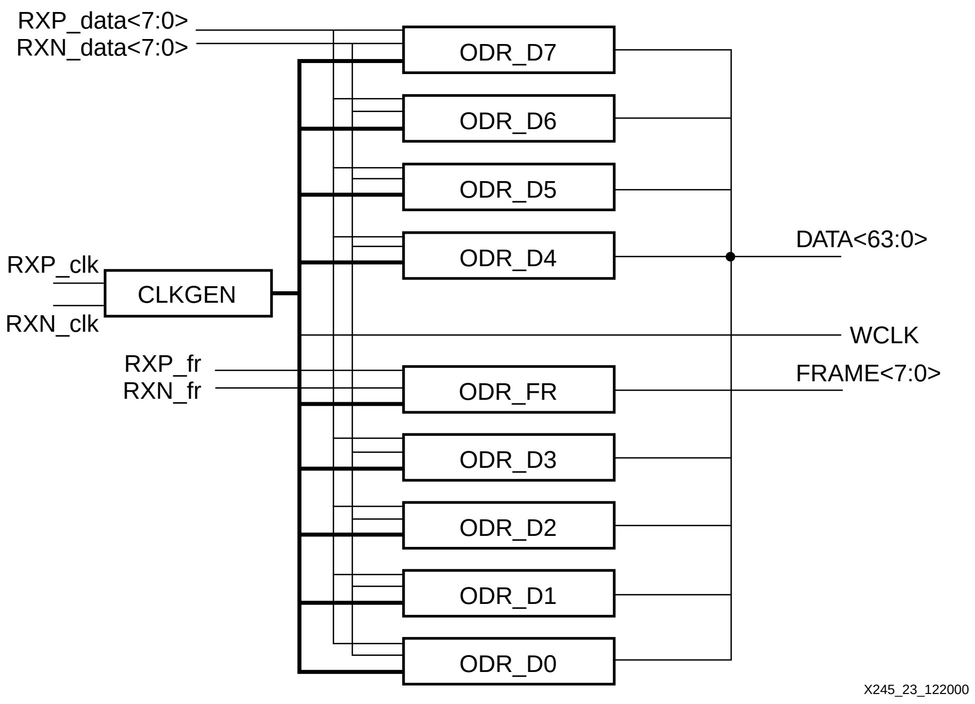


Figure 17: **HSRX Block Diagram**

CLKGEN Overview

Figure 18 shows the CLKGEN module. The CLKGEN module block divides the clock from the original input (CLK4X). Each clock is used on both the rising and falling edges. The maximum delay and skew budget increases with each division. Figure 19 shows the CLKGEN waveforms and how they are derived.The signal descriptions are shown in Table 8.

Table 8: **CLKGEN Module Pin Definitions**

I/O Type	Module Pin Name	Definition
Input	RXP, RXN	LVDS clock
	RST	Reset
Output	CLK4X	Receive clock (maximum 320 MHz)
	CLK2XR, CLK2XF	Half receive clock (maximum 160 MHz)
	CLK1XRR, CLK1XRF, CLK1XFR, CLK1XFF	Quarter receive clock (maximum 80 MHz)

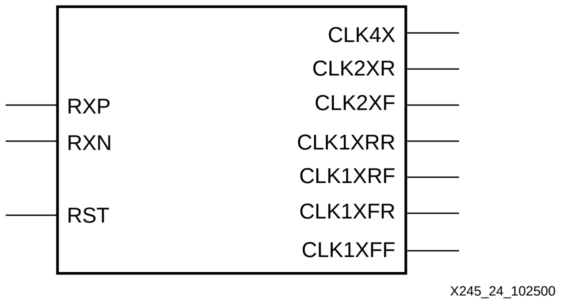


Figure 18: **CLKGEN Module**

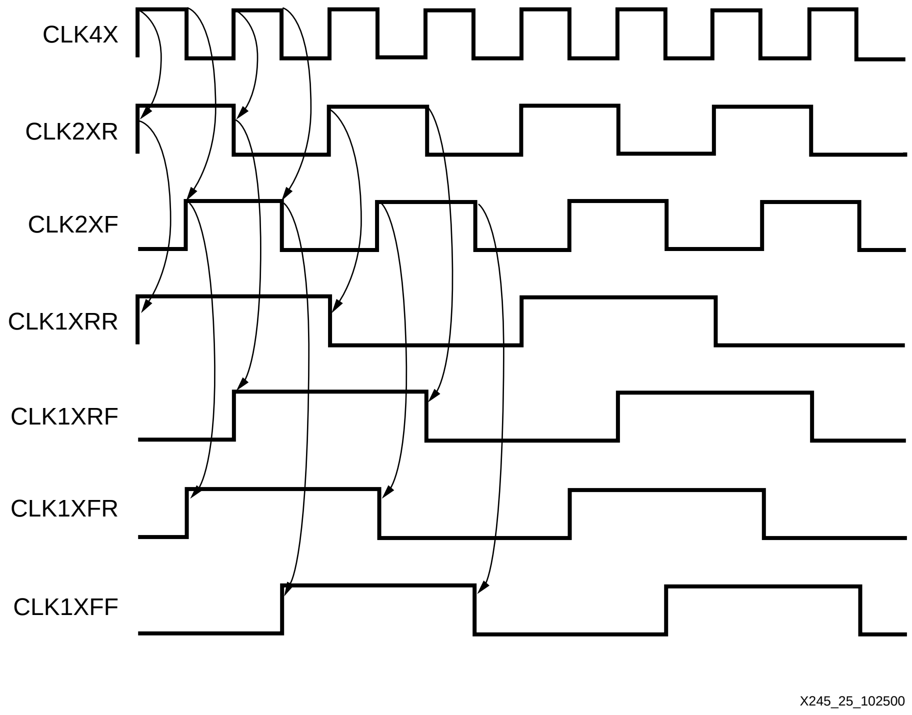


Figure 19: **CLKGEN Waveforms**

ODR_REG Overview

Figure 20 shows the Octal Data Rate Register (ODR_REG). The ODR_REG is a tree structure of DDR registers. The first register is a hard macro: DDRIN_left for the left side module and DDRIN_right for the right side module. It is required to guarantee timing from the IOB to the CLB registers. The DDR clock is always in the center of the data eye. Figure 21 shows the ODR_REG gate level diagram. Figure 22 shows the ODR_REG waveforms. The module pin descriptions are listed in Table 9.

Table 9: **ODR_R_REG Module Pin Definitions**

I/O Type	Module Pin Name	Definition
Input	RXP, RXN	LVDS data
	C4X	Receive clock (maximum 320 MHz)
	C2XR, C2XF	Half receive clock (maximum 160 MHz)
	C1XRR, C1XRF C1XFR, C1XFF	Quarter receive clock (maximum 80 MHz)
Output	Q<7:0>	Received data synchronous to the falling edge of CLK1XFR

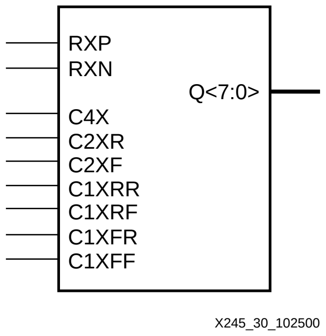
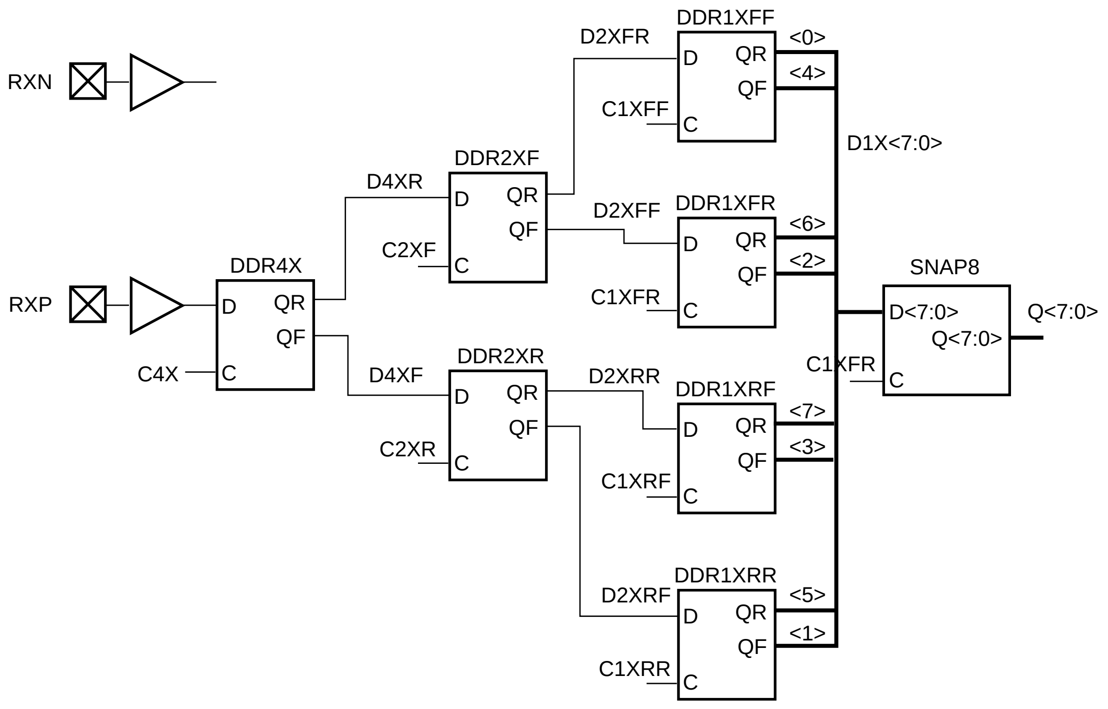
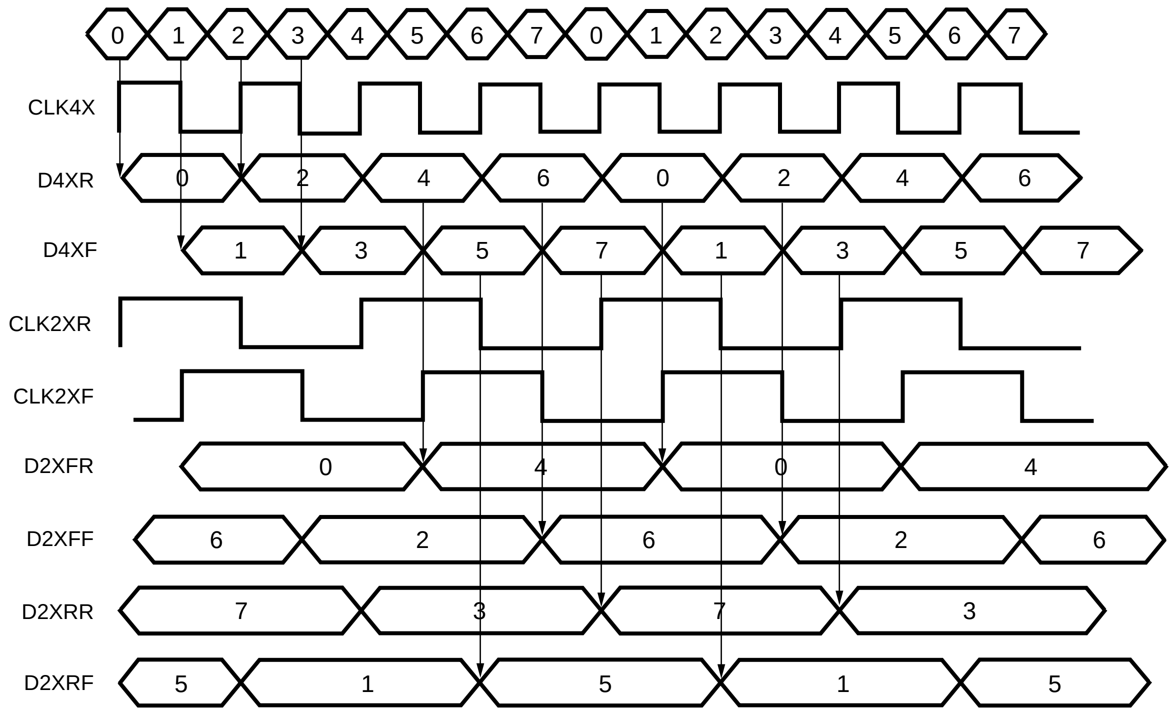


Figure 20: **ODR_REG Module**



X245_31_122000

Figure 21: **ODR_REG Gate Level Diagram**



X245_32_102500

Figure 22: **ODR_REG Waveform 1**

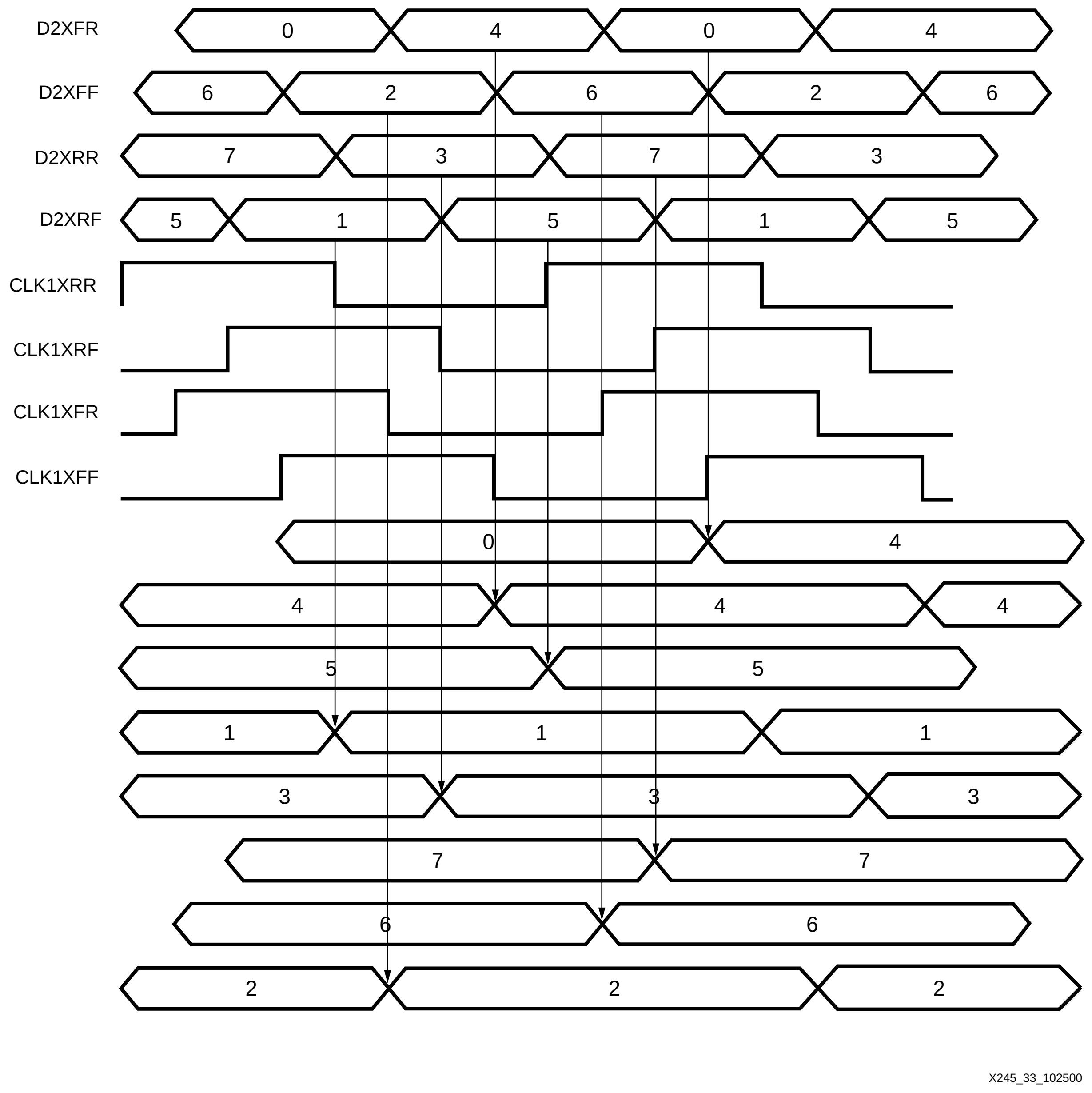


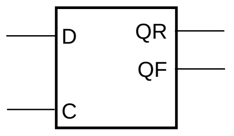
Figure 23: **ODR_REG Waveform 2**

DDREG Overview

Figure 24 shows the DDR register (DDREG). The DDREG is a simple two flip-flop circuit. It is used to split data into rising and falling edge data. Figure 25 shows the DDREG gate level diagram. The module pin descriptions are listed in Table 10.

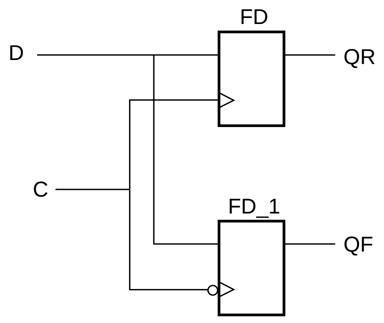
Table 10: **DDREG Module Pin Definitions**

I/O Type	Module Pin Name	Definition
Input	D	LVDS input
	C	CLK
Output	QR	Rising edge data
	QF	Falling edge data



X245_37_102500

Figure 24: **DDREG Macro**



X245_38_102500

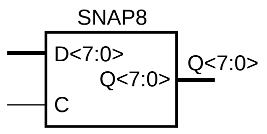
Figure 25: **DDREG Gate Level Diagram**

SNAP8 Overview

Figure 26 shows the 8-bit Snapshot (SNAP8). SNAP8 samples 8-bits from eight clock domains and presents it to a single clock domain. The first four bits are captured on the rising edge of C (CLK1XFR) and transferred to the falling edge. The second four bits are captured on the falling edge of C (CLK1XFR). The module pin descriptions are listed in Table 11.

Table 11: **SNAP8 Module Definitions**

I/O Type	Module Pin Name	Definition
Input	D <7:0>	Data
	C	CLK (80 MHz)
Output	Q <7:0>	Snapshot data



X245_39_102500

Figure 26: **SNAP8 Module**

FIFO Overview

Figure 27 shows the FIFO module. Used for crossing clock domains (RX and System) it is a simple buffer status bus showing the number of blocks of 16 64-bit words left in the FIFO. RST clears the read/write counters. The FIFO is 256 64-bit words deep. Figure 28 shows the FIFO block diagram. The module pin descriptions are listed in Table 12.

The control of the FIFO module is user definable. A typical control system waits until the buffer is near full (i.e., BUFSTAT = `1101`) before performing another read until the buffer is near empty (i.e., BUFSTAT = `0010`).

Table 12: **FIFO Module Pin Definitions**

I/O Type	Module Pin Name	Definition
Input	WDATA <63:0>	Write Data Bus
	WFRAME<7:0>	Write Frame Bus
	WCLK	Write CLK
	RCLK	Read CLK (System Clock)
	RE	Read Enable
	RST	FIFO Reset
Output	RDATA <63:0>	Read Data Bus
	RFRAME <7:0>	Read Frame Bus
	BUFSTAT<3:0>	Buffer Status

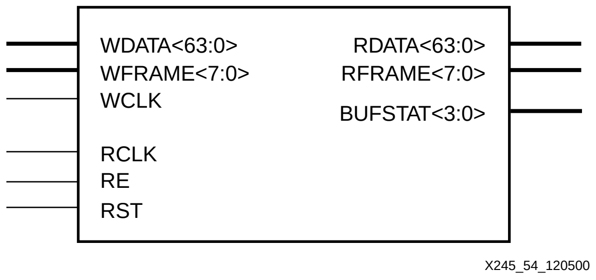


Figure 27: **FIFO Module**

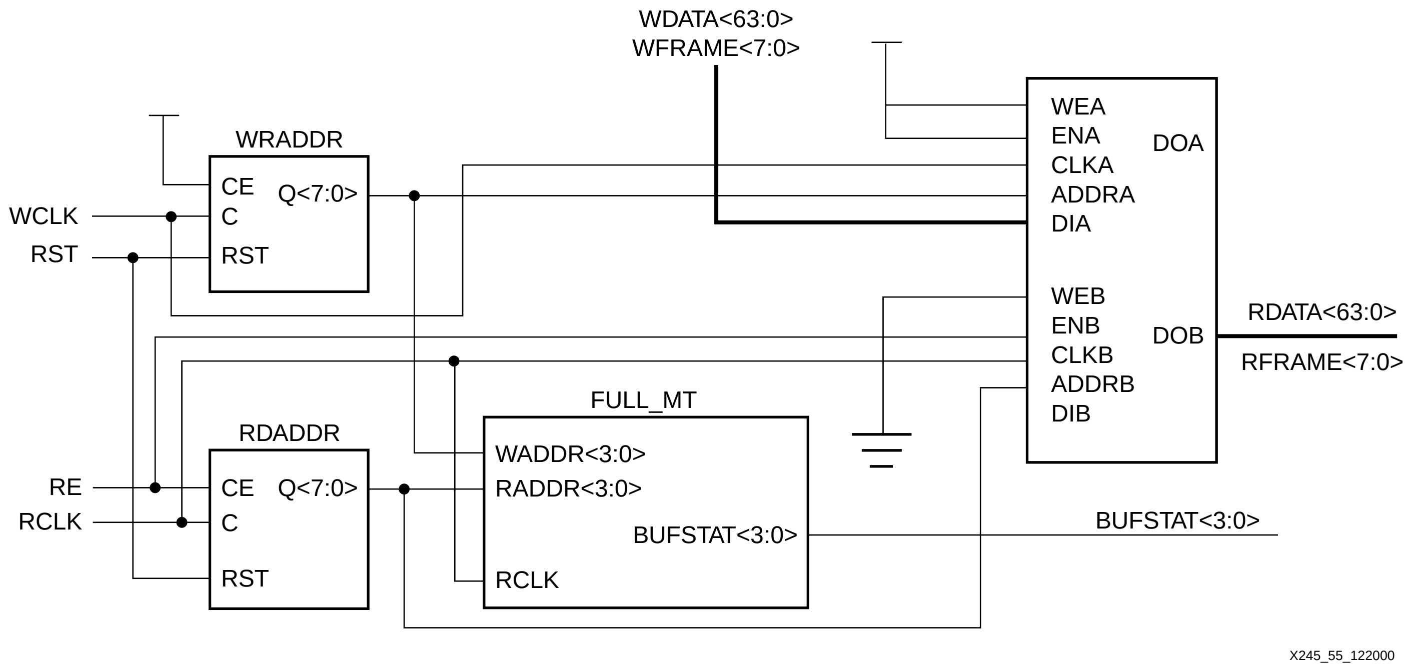


Figure 28: **FIFO Block Diagram**

ALIGN Overview

Figure 29 shows the ALIGN module. ALIGN takes raw data streams from the FIFO and aligns them to 64-bit word boundaries. The start of a boundary can be can be in one of four positions. The frame bus is used to determine the start of the word Table 13. The signal descriptions are shown in Table 14.

Table 13: **Frame Bus Positions**

Position	Word
Position 0	0000_1111
Position 1	0011_1100
Position 2	1111_0000
Position 3	1100_0000

Table 14: **ALIGN Module Pin Definitions**

I/O Type	Module Pin Name	Definition
Input	RDATA <63:0>	Raw data bus
	RFRAME <7:0>	Raw frame data
	C	CLK
Output	ADATA<63:0>	Aligned data bus
	AFRAME<7:0>	Aligned frame bus

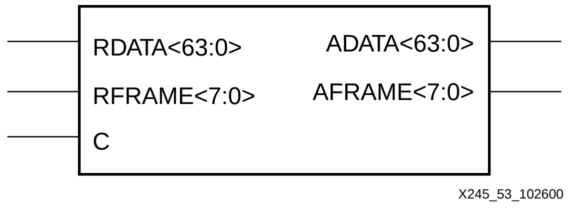


Figure 29: **ALIGN Module**

FULL_MT Overview

Figure 30 shows the FULL_MT module, a simple Full verses Empty generator. The FULL_MT module synchronizes the Write address to the Read clock domain. The buffer status is synchronous to the Read clock. Figure 28 shows the FULL_MT block diagram. The module pin descriptions are listed in Table 15.

Table 15: **FULL_MT Module Pin Definitions**

I/O Type	Module Pin Name	Definition
Input	WADDR <3:0>	Top 4 bits of write address bus
	RADDR<3:0>	Top 4 bits of read address bus
	RCLK	Read CLK
Output	BUFSTAT<3:0>	Buffer Status

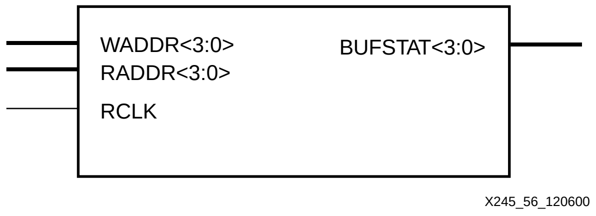


Figure 30: **FULL_MT Module**

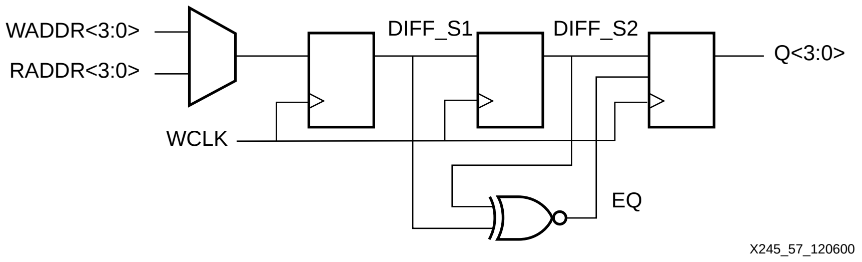


Figure 31: **FULL_MT Block Diagram**

Floorplan

The receiver does not require the use of the floorplan or guide files. The library of design files created for all possible locations for each part and package combination is available on the Xilinx web site (see **Using the Library of Designs**). Each data channel is implemented as an RPM macro and absolutely placed for the specific part, package, and locations.

The receiver is limited to 12 rows, ideally starting on a RAMB4 boundary row. Six rows above RXP_clk and six rows below RXP_clk. The clock and frame should be centered. The data channels are split evenly on either side. The data channels require the P-side location all to be in the same row (due to the IOB drives into the array).

The FIFO module requires five RAMB4s. Figure 32 shows the floorplan for a right side location in an XCV600E (FG676 package).

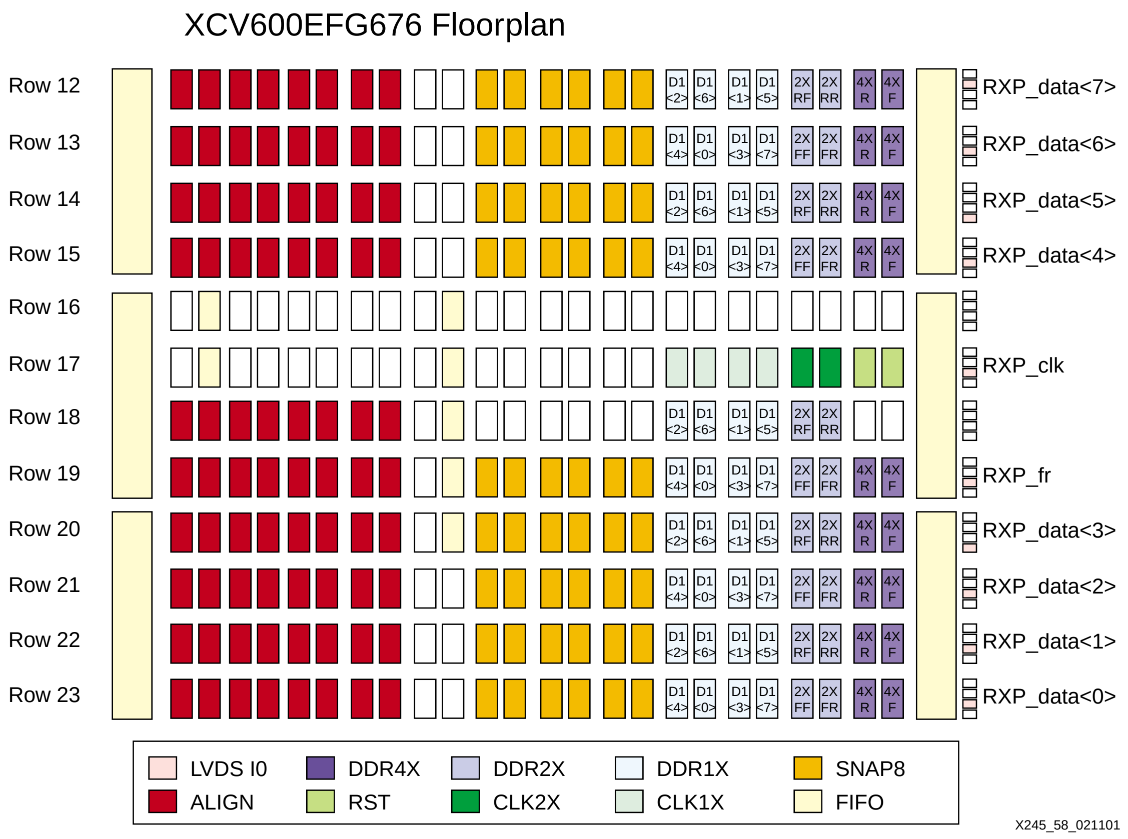


Figure 32: **Typical Receiver Floorplan**

Timing Issues

The route for the CLK4X net must use hex lines to hit each of the CLK inputs of the DDRIN modules.

Using the Library of Designs

A library of design files has been created for all possible locations for each part and package combination. The Verilog and VHDL files in the sim directory are for simulation only. Although the files are functionally accurate to the physical implementation, they are not synthesizable. Other than the standard IEEE libraries, the VHDL simulation files do not require any additional library files. When synthesizing the design, the cores are treated as black boxes. The I/O pads must not be inserted on the TX or RX LVDS ports.

The synth directories contain vendor specific instantiation code and example instantiation for both VHDL and Verilog. After synthesizing the full design using black box cores and generating an EDIF file, copy the specific core EDIF file and NMC file to the design directory. This allows NGDBUILD to find the necessary files when working with the black boxes.

The library design files can be found on the Xilinx web site at: "www.xilinx.com/products/virtex/techtopic/lvds/iostd/index.htm".

Synthesizing with Exemplar

To prevent I/O pad insertion on the LVDS input and output ports, attach a "nopad" attribute to the top level signal name. When synthesizing the Verilog code, one of the following file names must also be read. These files are not required for VHDL code. The design library includes a top level design example illustrating the use of the cores.

```
txlvds_1c_8d_1f _device_edge#_synth_exemplar.v
rxlvds_1c_8d_1f _device_edge#_synth_exemplar.v
```

Synthesizing with Synopsys FPGA Compiler

The Synopsys FPGA Compiler only inserts I/O pads on ports with a port_is_pad attribute. If one of the RX or TX ports has this attribute attached (by using the global set_port_is_pad command) then it can be removed using the remove_attribute {port_name} port_is _pad command. When synthesizing the Verilog code, one of the following file names must also be read. These files are not required for VHDL code. The design library includes a top level design example illustrating the use of the cores.

```
txlvds_1c_8d_1f _device_edge#_synth_synopsys.v
rxlvds_1c_8d_1f _device_edge#_synth_synopsys.v
```

Synthesizing with Synopsys FPGA Express

The Synopsys FPGA Express has an all or none function when inserting I/O pads. Designs using FPGA Express must include direct instantiation of I/O cells for all non-LVDS ports. The insert I/O option must be turned off. When synthesizing the Verilog code, one of the following file names must also be read. These files are not required for VHDL code. The design library includes a top level design example illustrating the use of the cores.

```
txlvds_1c_8d_1f _device_edge#_synth_synopsys.v
rxlvds_1c_8d_1f _device_edge#_synth_synopsys.v
```

Synthesizing with Synplicity

A "black_box_pin" attribute is required by Synplicity to prevent I/O pad insertion. Attach this attribute to the RX and TX module and the RX and TX LVDS pins. The design example illustrates this attachment in the HDL code. When synthesizing the Verilog code, one of the following file names must also be read. These files are not required for VHDL code. The design library includes a top level design example illustrating the use of the cores.

```
txlvds_1c_8d_1f _device_edge#_synth_synplicity.v
rxlvds_1c_8d_1f _device_edge#_synth_synplicity.v
```

Conclusion

The library of designs available on the Xilinx web site (see **Using the Library of Designs**) allows the designer to implement the fastest possible eight channel, one clock, one frame LVDS transmitters and receivers.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/02/01	1.0	Initial Xilinx release.
03/15/01	1.1	Minor revisions to Table 2 and Table 9.