XAPP607 (v1.0) April 17, 2002

Virtex-II Connection to a High-Speed Serial Device (TLK2501)

Author: Marc Defossez

Summary

This application note shows how to interface an external high-speed serial communications device to a Xilinx Virtex™-II FPGA. It also shows how the hardware inside the FPGA can be designed by means of a working example.

Introduction

This design is targeted towards the XLVDS demonstration board, but can be used in all Virtex-II devices. The XLVDS demonstration board contains a TLK2501IRCP device connected to an XC2V1000-5FG456 FPGA.

The demonstration board and this design can be used to evaluate the performance of a high-speed serial device connected to an FPGA. The demonstration board layout is designed and optimized to support high-speed operation. Thus, understanding impedance control and transmission line effects are crucial when designing high-speed boards:

- The printed-circuit board (PCB) is designed for high-speed signal integrity.
  - The board's impedance is controlled to 50W for both the high-speed differential serial and parallel data connections.
  - Impedance mismatches are reduced by designing the component pad size to be as close as possible to the width of the connecting transmission lines.
  - Vias are minimized and, when necessary, placed as close as possible to the device drivers.
  - The board contains both serial and parallel transmission lines and so care was taken to control both impedance and trace length mismatch (board skew).
- The PCB can be configured for copper or optical interfaces.
- SMB and parallel fixtures are easily connected to test equipment.
- On-board capacitors provide AC coupling of high-speed signals.

TLK2501 Transceiver

The TLK2501 transceiver is a member of a multi-gigabit family of transceivers used in ultra high-speed, bidirectional, point-to-point data transmission systems. This SerDes supports an effective serial interface speed of 1.6 Gb/s to 2.5 Gb/s, providing up to 2 Gb/s of data bandwidth. The component is a member of a complete family of high-speed serial devices, providing communication possibilities from 0.6 GHz up to 3.125 GHz. The other family members are:

- The TLK1501 a 0.6 Gb/s to 1.6 Gb/s transceiver
- The TLK3101 a 2.5 Gb/s to 3.125 Gb/s transceiver

The primary application of this chip is to provide high-speed data communication for point-to-point connections over controlled impedance media of approximately 50 ohms. The transmission media can be a printed-circuit board, copper cables, or fiber-optic cable. This device can also be used to replace parallel data transmission architectures by providing a reduction in the number of traces, connector terminals, and transmit/receive terminals.

Parallel data loaded into the transmitter is delivered to the receiver over a serial channel and is then reconstructed into its original parallel format.

The TLK2501 transceiver performs data conversion parallel-to-serial and serial-to-parallel:

- The transmitter latches 16-bit parallel on the supplied reference clock (GTX_CLK). The 16-bit parallel data is internally encoded into 20 bits using an 8-bit/10-bit (8B/10B) encoding format. The resulting 20-bit word is then transmitted via differential outputs at 20 times the reference clock rate.
- The receiver section performs the serial-to-parallel conversion on the input data, synchronizing the resulting 20-bit wide parallel data to the extracted reference clock (RX_CLK). It then decodes the 20-bit wide data using 8-bit/10-bit decoding format, resulting in 16 bits of parallel data and it present this to the receive data pins (RXD0-15).

The outcome is an effective data payload of 1.28 Gb/s to 2.0 Gb/s (16 bits data x the GTX_CLK frequency).

Figure 1 is a block diagram of a TLK high-speed SerDes that can be used throughout the family of devices.

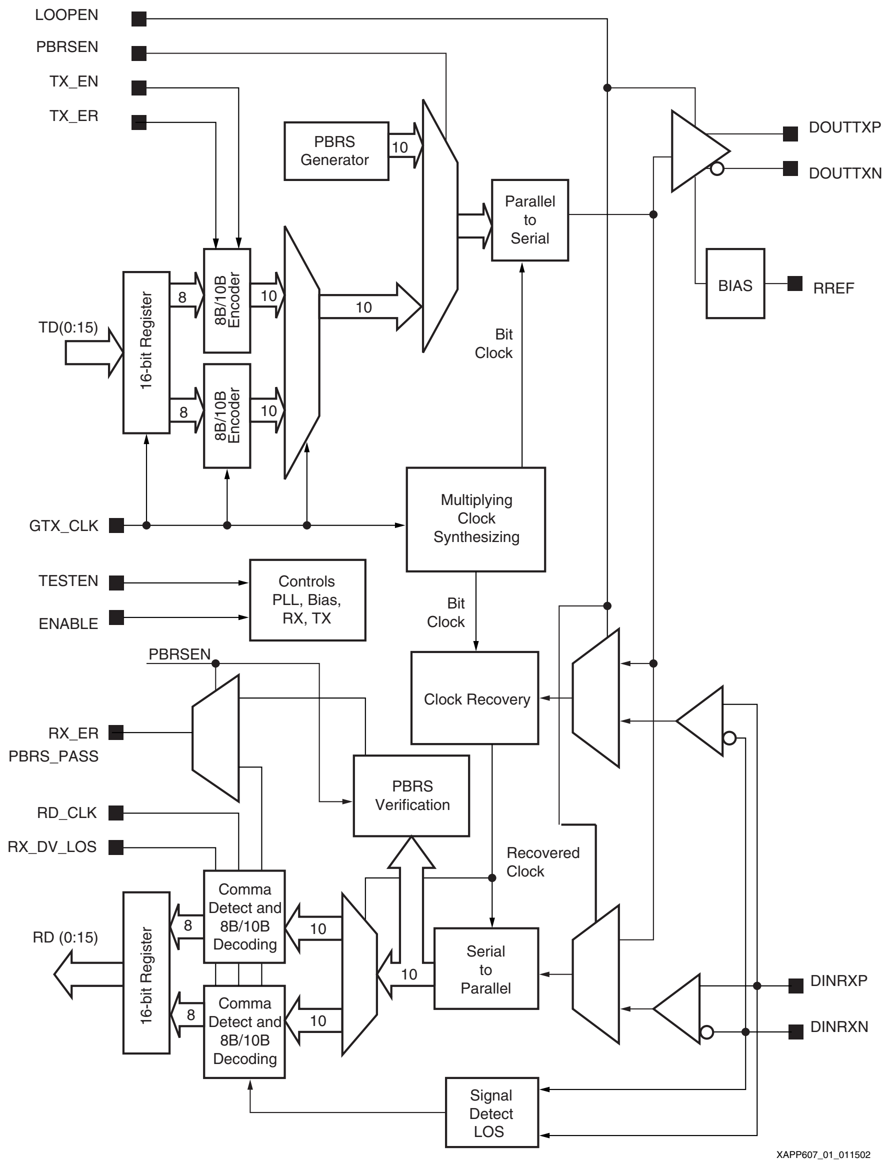

Figure 1: **TLK2501 Block Diagram**

Transmitter Interface

Transmit Parallel Data Bus Interface

The transmit bus interface connected to the FPGA accepts 16-bit single-ended TTL parallel data at the TXD [0:15] terminals. Data is valid on the rising edge of the GTX_CLK when the TX_EN is High and the TX_ER is Low. The GTX_CLK is used as the word clock.

The data, enable, and clock signals must be properly aligned as shown in Figure 2.

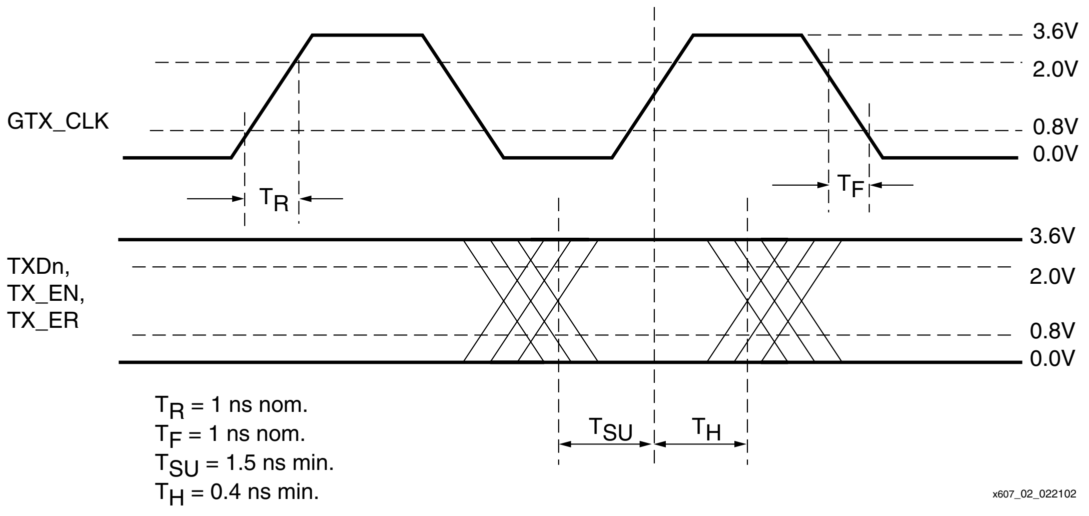

Figure 2: **Transmit Waveform**

Transmitter

The transmitter section validates incoming 16-bit wide data (TXD [0:15]) on the rising edge of the GTX_CLK. That incoming data is then 8-bit/10-bit encoded, serialized, and transmitted sequentially over the differential high-speed I/O channel. The clock multiplier multiplies the reference clock (GTX_CLK) by a factor of 10, creating a bit clock. This internal bit clock is fed to the parallel-to-serial shift register. This register transmits data on both edges of the bit clock, providing serial data at a rate of 20 times the reference clock. Data is transmitted LSB (D0) first.

Due to the parallel-to-serial conversion and clock multiplication, a latency effect exists between the moment data is loaded from the parallel bus into the transmit input register and the moment the first serial bit 0 is present at the differential outputs. This effect is called data transmission latency.

The minimum transmit latency ($T_{latency}$) is 34 bit times; the maximum is 38 bit times. Figure 3 illustrates the timing relationship between the transmit data bus; the GTX_CLK and serial transmit terminals.

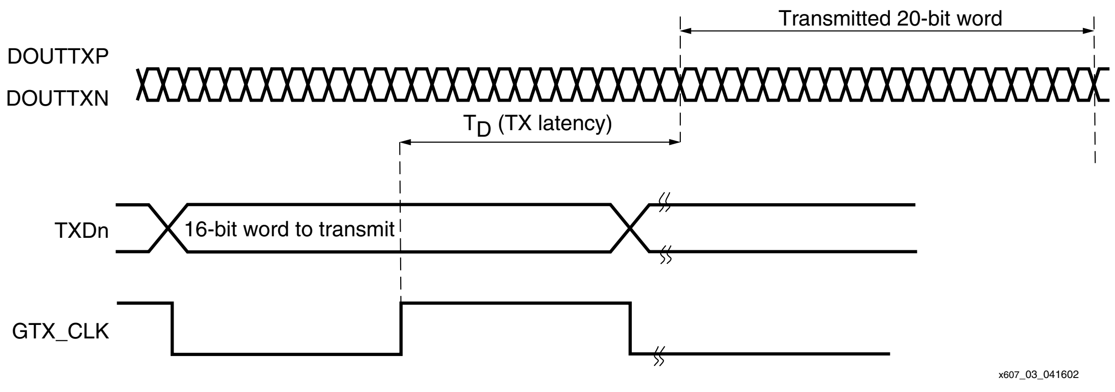

Figure 3: **Transmit Latency**

Transmitter 8-Bit/10-Bit Encoding

An 8-bit/10-bit encoding algorithm is implemented in the transmitter section of the device; it's the same algorithm used by fiber channel and the gigabit Ethernet. The decoding is transparent to the user; data is internally encoded such that the user only has to write 16-bit data.

The 8-bit/10-bit encoder converts 8-bit wide incoming data to a 10-bit wide encoded data character. This is done to improve its transmission characteristics. The encoding scheme maintains the signal DC balance by keeping the number of ones and zeros the same. This provides good transition density for clock recovery and improves error checking.

Externally, the device has a 16-bit wide interface; internally, the data is split into two 8-bit wide bytes for encoding. Each byte is fed into a separate encoder. The encoding is dependent upon two additional input signals, the TX_EN and TX_ER. Table 1 provides the transmit data control decoding.

Table 1: **TX Data Control**

TX_EN	TX_ER	Encoded 20-Bit Output
0	0	IDLE (<K28.5, D5.6> or <K28.5, D16.2>)
0	1	Carrier extend (K23.7, K23.7)
1	0	Normal Data Character
1	1	Transmit error propagation (K30.7, K30.7)

Since the data is transmitted in 20-bit serial words, K codes indicating carrier extend and transmit error propagation are transmitted as two 10-bit K-codes. When no payload data is available, the encoder inserts the IDLE character. IDLE consists of a K28.5 (BC) code and either a D5.6 (C5) or a D16.2 (50) character. The IEEE 802.3z specification defines the K28.5 character as a pattern consisting of `011111010` with the 7 MSBs (`0011111`) referred to as the comma character. IDLE consists of two 10-bit codes, 20-bits wide, that are transmitted during a single GTX_CLK cycle.

Receiver Interface

Receiver Parallel Data Bus

The receive bus drives 16-bit wide, single-ended, TTL parallel data at the RXD [0:15] terminals. Data is valid on the rising edge of the RX_CLK when the RX_DV/LOS is High and the RX_ER is Low. The RX_CLK is used as the recovered word clock. The data, enable, and clock signals are aligned as shown in Figure 4.

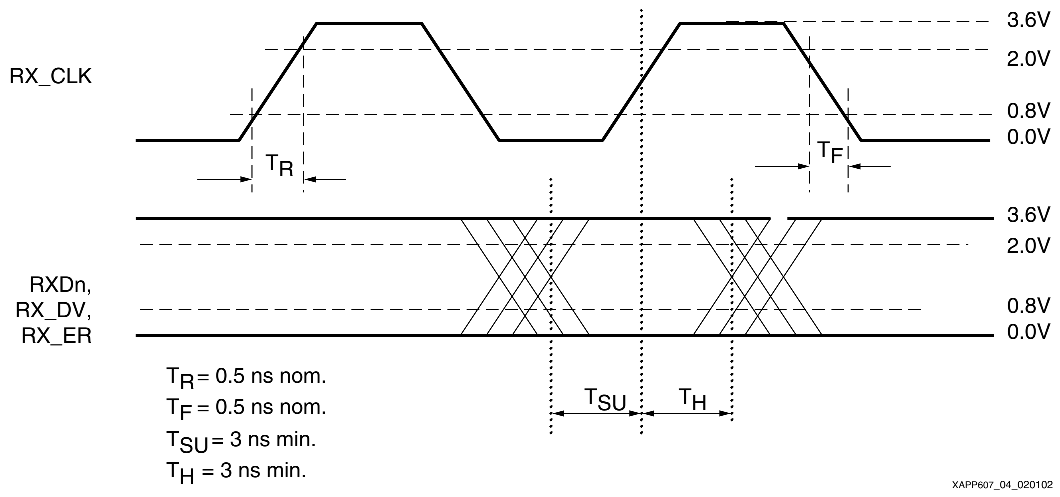

Figure 4: **Receive Waveform**

Receiver

The receiver section accepts 8-bit/10-bit encoded differential serial data. The interpolator and clock recovery circuit lock to the data stream and extract the bit rate clock. This recovered clock is used to re-time the input data stream. The serial data is then clocked into the serial-to-parallel shift registers. The 10-bit wide parallel data is then multiplexed and fed into two separate 8-bit/10-bit decoders, where the data is then synchronized to the incoming data steam word boundary by detection of the K28.5 synchronization pattern.

Receive latency is the time between the incoming data at the differential terminals and the presentation of a 16-bit parallel output. The receive latency is fixed after the link is established. The minimum receive latency ($R_{latency}$) is 76 bit times; the maximum is 107 bit times. Figure 5 illustrates the timing relationship between the serial receive terminals, the recovered word clock (RX_CLK), and the receive data bus.

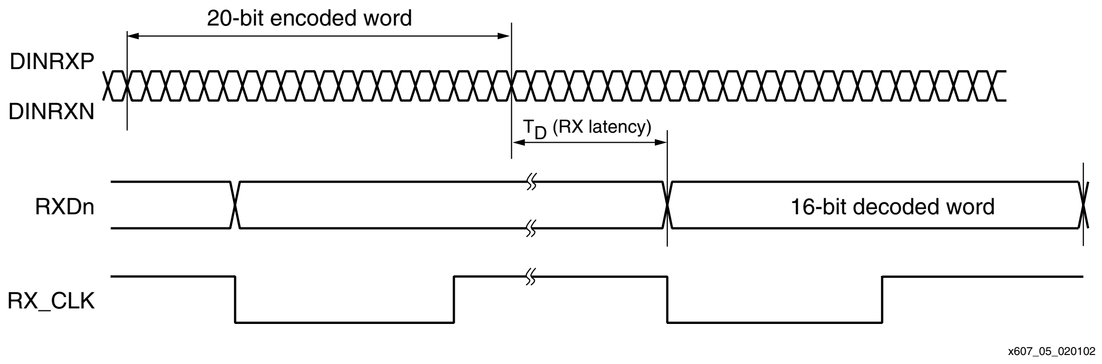

Figure 5: **Receive Latency**

Receiver Comma Detect and 8-bit/10-bit Decoding

Incoming data is serial-to-parallel converted and then presented at two 8-bit/10-bit decode circuits. Each 8-bit/10-bit decoder converts 10-bit encoded data back into 8-bits. The comma detect circuit provides byte synchronization to an 8-bit/10-bit transmission code. When the serial data is received and converted to parallel format, a way is needed to recognize the byte boundary. Generally, this is accomplished with a synchronization pattern. This is generally a unique pattern of 1s and 0s that either cannot occur as part of the valid data or it's a pattern that repeats at defined intervals. 8-bit/10-bit encoding contains a character called the comma (`b0011111` or `b1100000`), which is used by the comma detect circuit to align the received serial data back to its original byte boundary.

The decoder detects the K28.5 comma, generating a synchronization signal aligning the data to their 10-bit boundaries for decoding. It then converts the data back into 8-bit data, removing the control words. The output from the two decoders is latched into the 16-bit register synchronized to the recovered parallel data clock (RX_CLK) and output valid on the rising edge of the RX_CLK.

Two output signals, RX_DV/LOS and RX_ER, are generated along with the decoded 16-bit data output on the RXD [0:15] terminals. The output status signals are asserted as shown in the Table 2.

Table 2: **RX Data Control**

Received 20-Bit Data	RX_DV/LOS	RX_ER
Carrier extend (K23.7, K23.7)	0	1
IDLE (<K28.5, D5.6> or <K28.5, D16.2>)	0	0
Normal data character (Dx.y)	1	0
Receive error propagation (K30.7, K30.7)	1	1

If the decoded data is not a valid 8-bit/10-bit code, an error is reported by the assertion of both RX_DV/LOS and RX_ER. If the error was due to an error propagation code, the RXD bits output hex FEFE. If the error was due to an invalid pattern, the data output on RXD is undefined.

When an IDLE code decoded, both RX_DV/LOS and RX_ER are Low and a K28.5 (BC) code followed by either a D5.6 (C5) or D16.2 (50) code are output on the RXD terminals.

Additional Features

The SerDes device is equipped with several extra features, which are explained briefly below.

Loss of Signal Detection

A loss-of-signal-detection circuit is provided for conditions where the incoming signal has no longer sufficient voltage to keep the clock recovery circuit in lock.

The signal detection circuit is intended to be an indication of gross signal error conditions, such as a detached cable or no signal being transmitted, and not an indication of signal coding health.

The Loss of Signal (LOS) condition is reported by asserting, the RX_DV/LOS, RX_ER and RXD [0:15] all to a High state. As long as the incoming signal is above 200 mV in differential magnitude, the LOS circuit does not signal an error condition.

Power Down Mode

When the ENABLE terminal is Low, the SerDes device goes into a power down mode.

In power down mode, the serial transmit terminals (OUTTXP, DOUTTXN), the receive data bus terminals (RXD [0:15]), and the RX_ER are all placed high-impedance state.

The signal detection circuit stays active and when a valid differential signal amplitude of >200 mV on the serial receive terminals is present the RX_DV/LOS is driven Low.

In power down mode the GTX_CLK, clock signal must be provided.

PRBS Verification

The TLK2501 also has a built-in BERT function in the receiver side that is enabled by the PRBSEN. It can check for errors and report the errors by forcing the RX_ER/PRBSPASS terminal Low.

Loop Back Testing

The transceiver can provide a self-test function by enabling (LOOPEN) the internal loop back path. Enabling this terminal causes serial-transmitted data to be routed internally to the receiver. The parallel data output can be compared to the parallel input data for functional verification. (The external differential output is held in a high-impedance state during the loop back testing.) Combining this Loop back functionality with the PRBS function results in a possible very effective test of a high-speed serial data link.

Initialization and Synchronization

The TLK2501 has a synchronization state machine, which is responsible for handling link initialization and synchronization. See Figure 6.

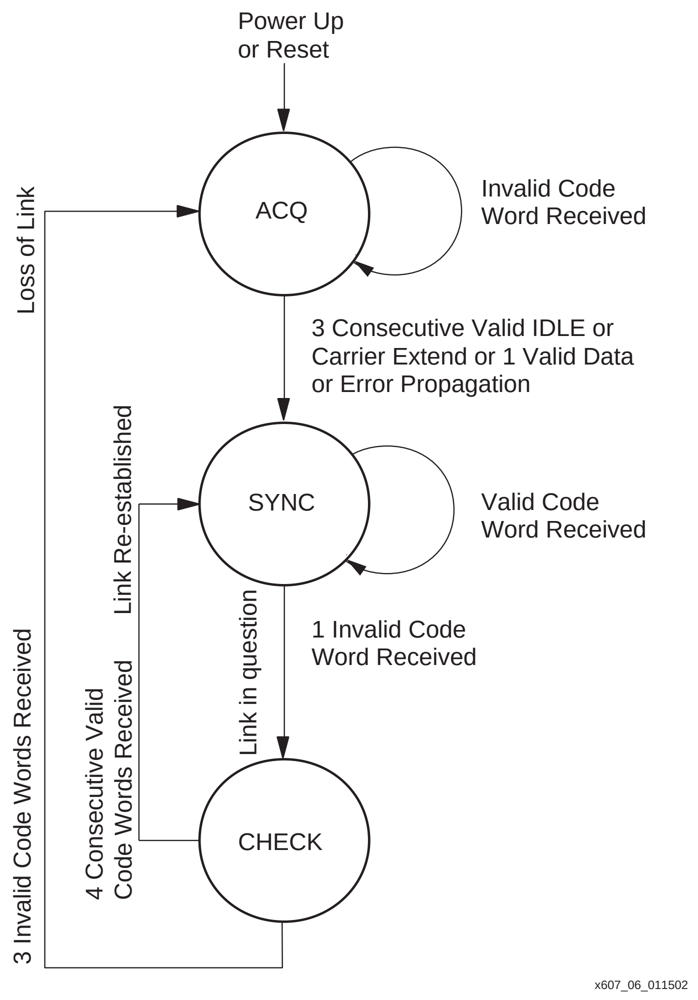
Power Up or Reset
ACQ
Invalid Code Word Received
3 Consecutive Valid IDLE or Carrier Extend or 1 Valid Data or Error Propagation
SYNC
Valid Code Word Received
1 Invalid Code Word Received
Link in question
CHECK
Link Re-established
4 Consecutive Valid Code Words Received
Loss of Link
3 Invalid Code Words Received
x607_06_011502

Figure 6: **Initialization and Synchronization**

Until power up or reset, the state machine enters the acquisition (ACQ) state and searches for IDLE. Upon receiving three consecutive IDLEs or a carrier extend, the state machine enters the synchronization (SYNC) state. If, during the acquisition process, the state machine receives valid data or an error propagation code, it immediately transitions to the SYNC state. The SYNC state is the state for normal device transmission and reception.

If during normal transmission and reception invalid codes are received, the synchronization state machine transitions to the CHECK state. The CHECK state determines whether the invalid code received was caused by a spurious event or a loss of the link. If in the CHECK state, the decoder sees four consecutive valid codes, the state machine determines the link is good and transitions back to the SYNC state for normal operation. If in the CHECK state, the decoder sees three invalid codes (not required to be consecutive), the device determines a loss of the link has occurred and transitions the synchronization state machine back to the link acquisition state (ACQ). The state of the transmit data bus, control terminals, and serial outputs during the link acquisition process is shown in Figure 7.

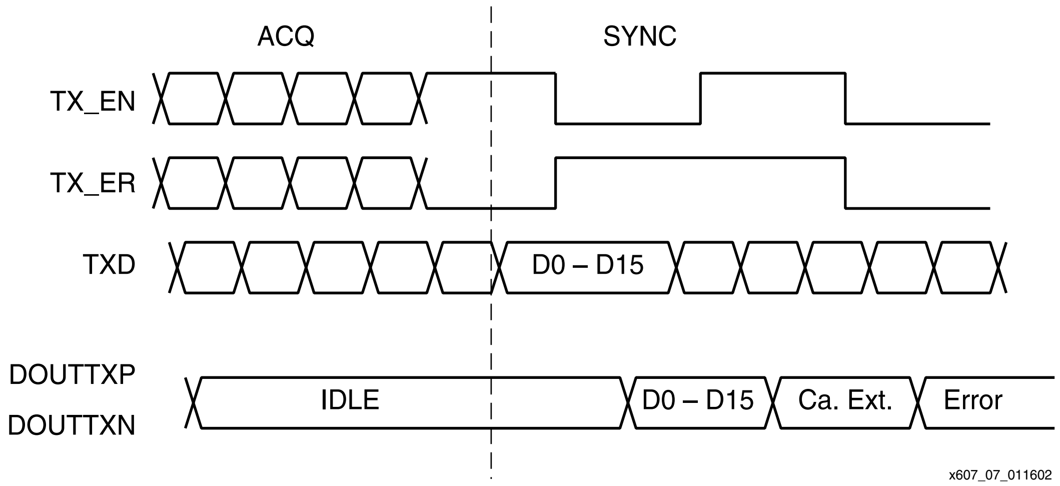

Figure 7: **TX Timing Diagram**

The state of the receive data bus, status terminals, and serial inputs during the link acquisition process is shown in Figure 8.

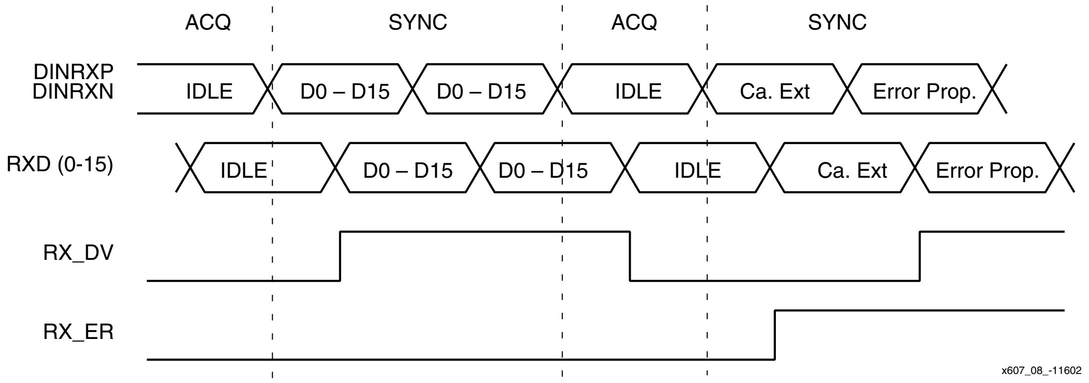

Figure 8: **RX Timing Diagram**

Virtex-II Devices

Notes:

1. An in-depth understanding of the Xilinx FPGA architecture is a prerequisite for using this application note and reference design.

In this reference design/application note, the transmitter and receiver parallel data buses are each connected to separate I/O banks of the FPGA. The transmitter and receiver control pins use the same bank as the parallel transmit bus. In this way, it is possible to get complete control of the high-speed serial device.

The TLK2501 has a core voltage of 2.5V. The transmitter input side can accept signals of LVTTL-level, and the receiver output bus generates signal levels accepted by normal LVTTL inputs from other devices. This makes it possible to use LVTTL standards for both TXD and RXD connections at the FPGA side.

Reference Design

General

The design is set up as a set of three registers to control the device, send data to, or receive data from the high-speed serial device. This approach makes it easy to use this design, with a little bit of extra, as peripheral for a micro-controller, such as Microblaze.

The possible design is given in Figure 9. The items marked in <red> are not part of this application note, but they are used to test the design.

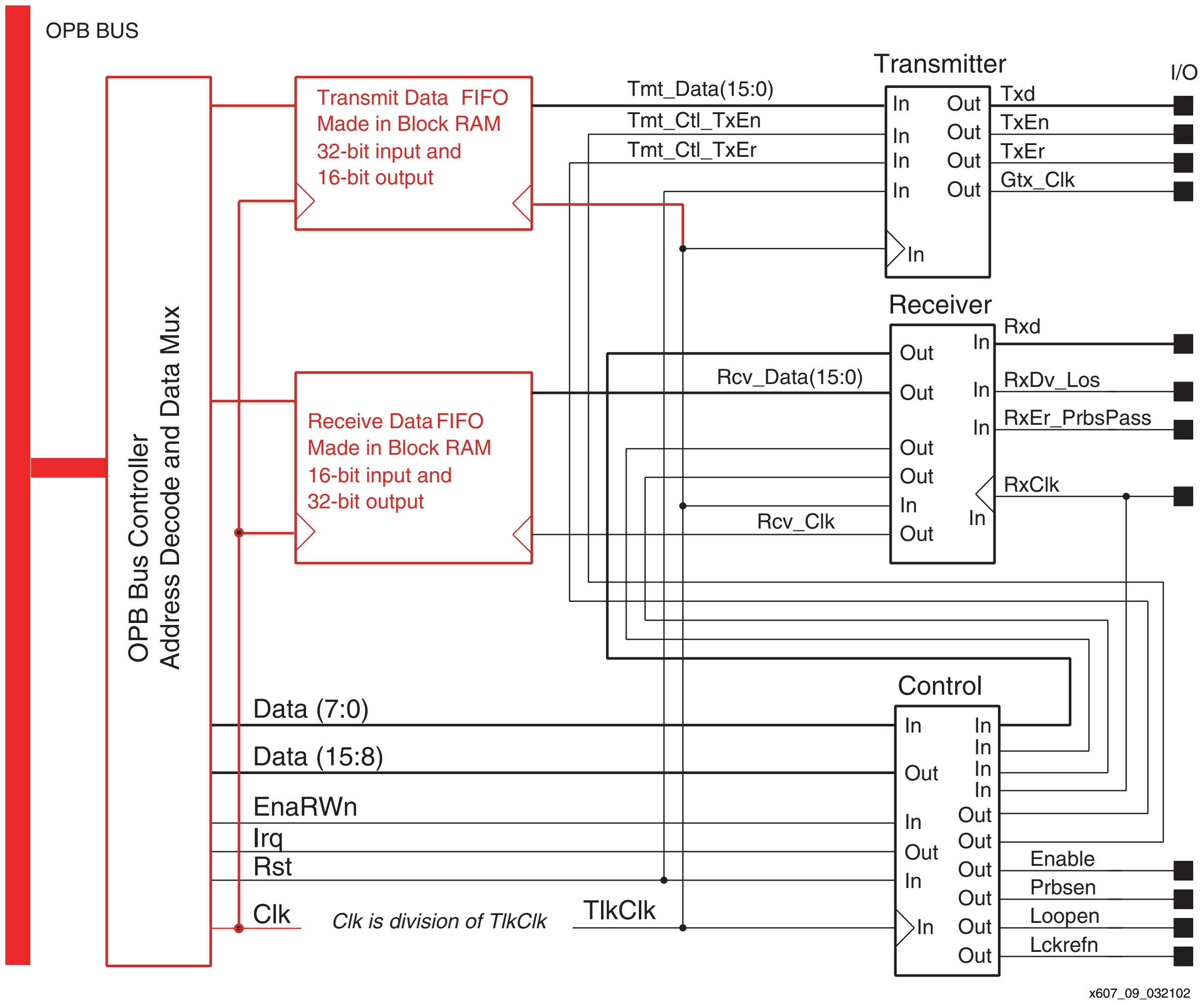

Figure 9: **FPGA-TLK Design**

The connections from the SerDes device to the FPGA are given in Figure 10.

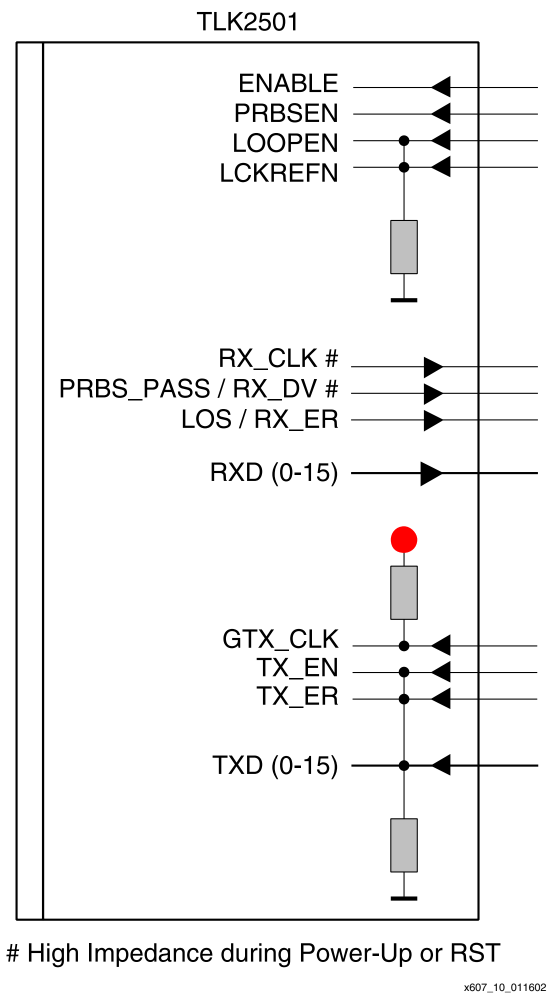

Figure 10: **TLK2501 Ports**

Top Level

The top level of the design contains all possible connections to and from the high-speed serial device. Note that the FPGA pin information will change depending your design.

Control Section

This sublevel of the design contains the control and status registers. An 8-bit write register and an 8-bit read register are also designed. See Figure 11 for the Control register.

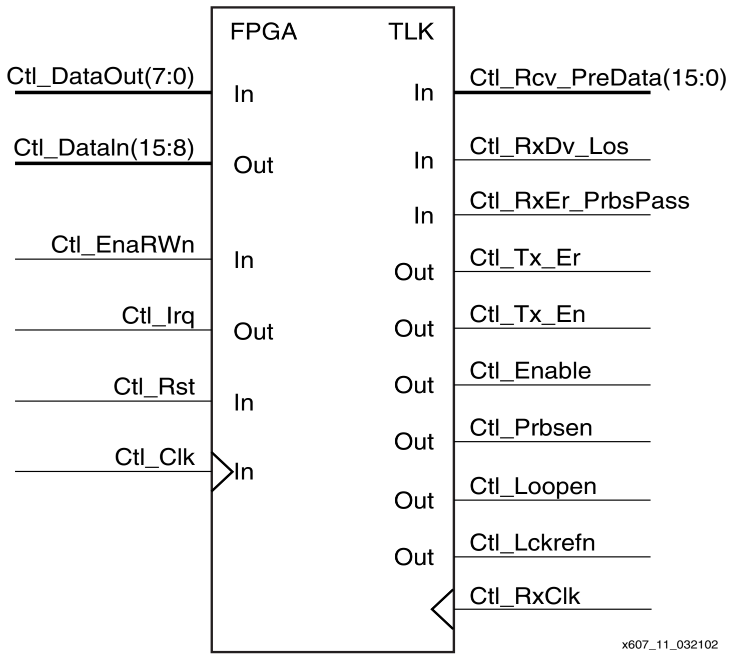

Figure 11: **Control**

A write operation is performed when Ctl_EnaRWn is pulled Low. A read operation can take place when Ctl_EnaRWn is High. The normal status of this input is High, so that read operations can take place without special actions.

The read and write data buses is combined into one 16-bit bus. The high byte of the bus is the read byte, while the low byte can be the write byte.

Control Write Register

This register controls the (nearly) static control pins of the SerDes and contains the two transmitter status control pins. See Table 3.

Table 3: **Control Write Register**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Enable	Prbsen	Loopen	Lckrefn	No use	No use	Tx_Er	Tx_En

Control Read Register

This register contains all status information coming from the high-speed SerDes device. When one of the status bits values changes, an interrupt is generated.

The registers are *not* cleared when a read operation is performed. A warning, under the form of a interrupt, is given each time a register bit status is changed, and then appropriate action can be taken. See Table 4.

Table 4: **Control Read Register**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PrbsPass	CableDet	Los	Loopset	CarExt	ErrProp	TxDv	Rx_Er

RXDv_Los and RxEr_PrbsPass connections are decoded and stored in a register. The decoding is set up as follows:

- When Enable is Low, the device is in a power down mode. Only the RxDv_Los net is functional. A high level on this pin indicates that the incoming differential signal is not sufficient for valid reception (a disconnected cable or bad connection can be the cause). These two statuses are used to create the CableDet signal.
- Only this bit of the read register can change, all other bits will remain zero.
- When Enable is High, the device is put into a full working state, and all bits of the read register can change depending the status of the control lines.
- When PRBSEN is brought High, the SerDes device enters a state in which it creates Pseudo Random Bitstreams. Then Bit PrbsPass will indicate is a valid pseudo random sequence is received.
- When during normal operation, the device no longer has sufficient differential input signal levels (ex. cable disconnect), this is indicated with the LOS bit. LOS is the decoding of Rx_Er (1), Rx_Dv (1) and Rxd (FFFF).
- CarExt (Carrier Extend) and ErrProp (Error Propagation) are a reflection of the decoded outputs of the TLK device.

- Rx_Er = 1, Rx_Dv = 1, Rxd = FEFE	reflects ErrProp
- Rx_Er = 1, Rx_Dv = 1, Rxd = F7F7	reflects CarExt

- When the LOOPEN function is invoked, by setting the LOOPEN bit of the write register, this is indicated to the controlling device with the LOOPSET bit.
- The other normal functional status is indicated with the RxDv and RxEr bits.

Notes:

1. When a "valid data" status is indicated, the received bits are directly put towards the receiver data bus, while the status is indicated in this read register. When a different status is indicated, the receiver data bus remains unchanged, while the status can be decoded and action can be taken.

Each time one of the read register bits changes state, an interrupt is generated. This interrupt is one clock period long and does not need a clear operation.

Figure 12 shows how the interrupt register is designed.

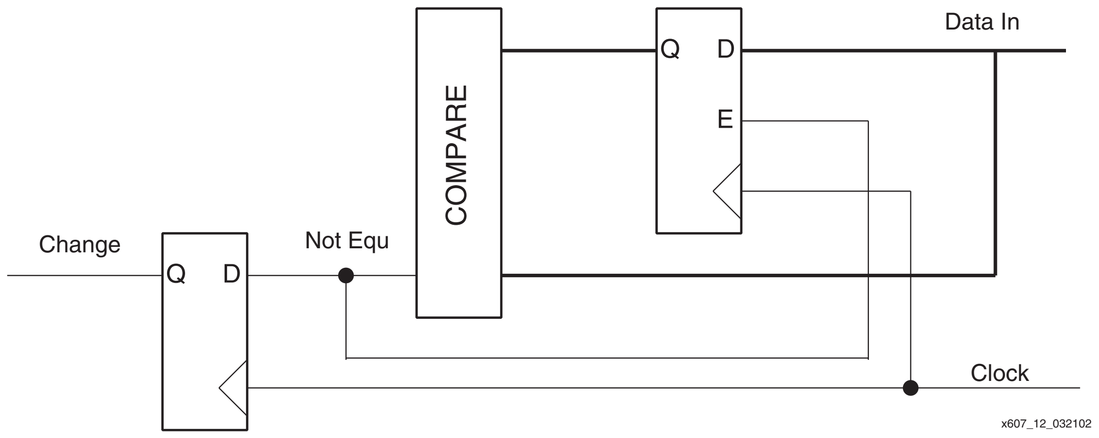

Figure 12: **Interrupt Generator Register**

Transmitter Section

Figure 13 is a diagram of the Transmit Register.

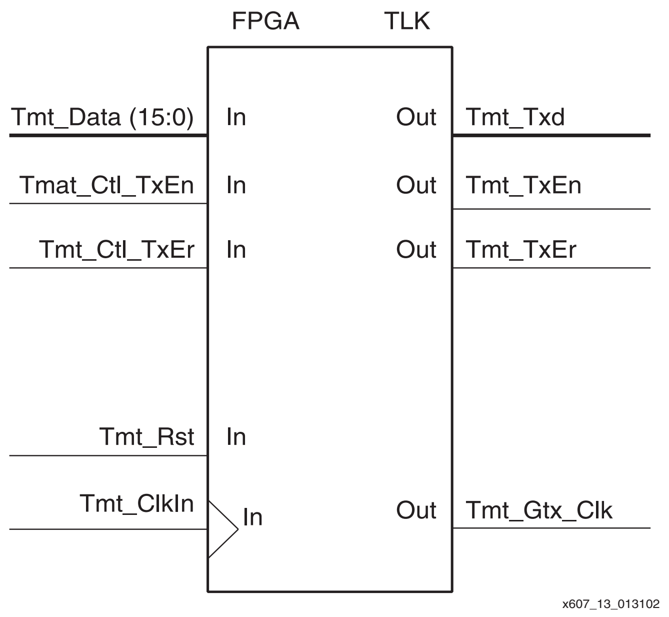

Figure 13: **Transmit Register**

Clock

The FPGA delivers the GTX_CLK clock and the data to the SerDes device. An oscillator of 125 MHz is needed for a serial transmission rate of 2.5 GHz. That oscillator must be connected to a clock input pin of the FPGA. Because there does not need to be an exact relationship between the incoming clock and the GTX_CLK clock for the SerDes device, no Digital Clock Manager (DCM) is needed. A DCM can be used when a more complex back-end design is added. When using a DCM in the transmitter section of a design, do not use a FPGA generated clock for the SerDes device. Instead, route on the PCB the GTX_CLK clock from the oscillator to FPGA and TLK devices, carefully matching the traces on the PCB.

See Figure 14 for GTX_CLK generation.

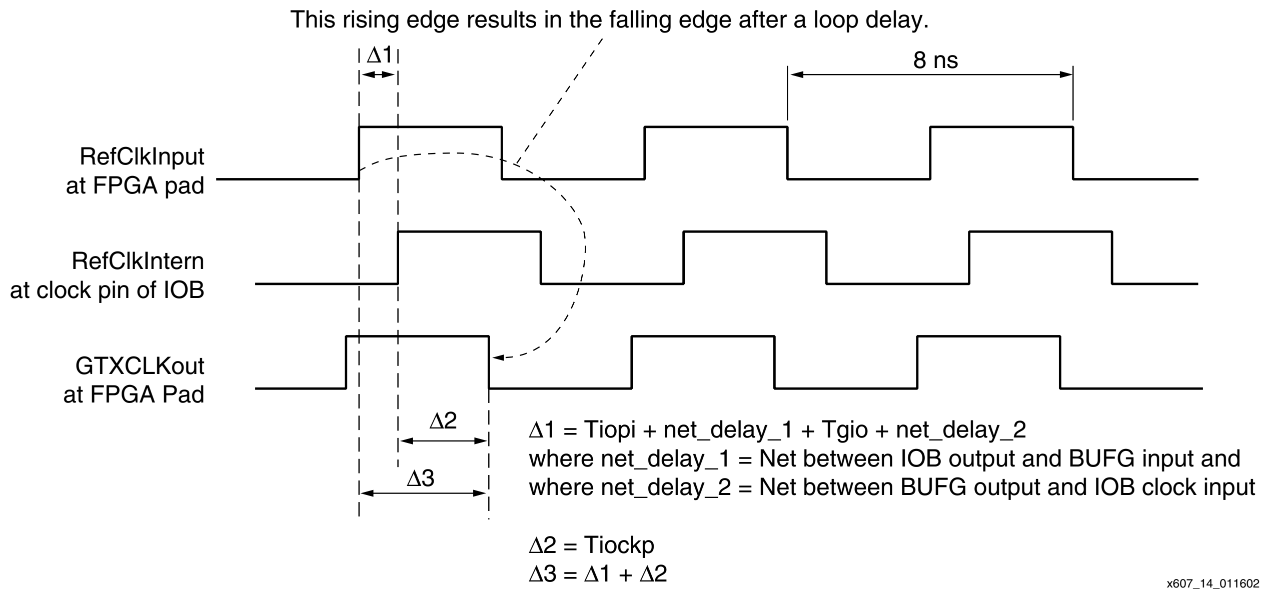
This rising edge results in the falling edge after a loop delay.
Δ1
8 ns
RefClkInput at FPGA pad
RefClkIntern at clock pin of IOB
GTXCLKout at FPGA Pad
Δ2
Δ3
Δ1 = Tiopi + net_delay_1 + Tgio + net_delay_2
where net_delay_1 = Net between IOB output and BUFG input and
where net_delay_2 = Net between BUFG output and IOB clock input
Δ2 = Tiockp
Δ3 = Δ1 + Δ2
x607_14_011602

Figure 14: **GTX_CLK Generation**

The clock is taken into the FPGA via a clock input pin. The GTX_CLK clock is generated by means of a Double Data Rate (DDR) flip-flop in the Input Output Block (IOB). Both flip-flops of the IOB are clocked with the reference clock of 125 MHz. The input of flip-flop FF0 of the DDR IOB must be connected to ground, while the input of flip-flop FF1 must be connected to a logic-one level.

The resulting output clock will be 180 degrees shifted in phase with the incoming reference clock. A delay between the incoming and outgoing clock exists due to the different delays added when routing through the FPGA.

An example of the accumulated timing follows:

```
Source:                  ClkIn
  Destination:             ClkOutFf/FF1
    Delay type          Delay(ns)  Logical Resource(s)
    ----------------------------  -------------------
    Tiopi                   0.822  ClkIn_ibuf/IBUFGPad to I
    net (fanout=1)          0.158  ClkIn_ibuf/IBUFG_netnet delay
    Tgi0o                   0.182  ClkIn_ibuf/BUFGClock buffer
    net (fanout=4)          0.902  ClkIn_c_netnet delay to FFs
    ----------------------------  ------------------------------
    Total                   2.064 ns (1.004 ns logic, 1.060 ns route)
  Source:                  ClkOutFf/FF1
  Destination:             ClkOut
    Delay type          Delay(ns)  Logical Resource(s)
    ----------------------------  -------------------
    Tiockp                  2.811  ClkOutFf/FF1FF clock to pad
    ----------------------------  ------------------------------
    Total                   2.811 ns (2.811 ns logic, 0.000 ns route)
```

Total accumulated delay between input and output is: 2.064 + 2.811 = 4.875 ns.

Make sure that the DDR flip-flops generating the CTX_CLK clock are always enabled; even when the TLK SerDes is in power down mode, the GTX_CLK clock must be available.

Data

The data to transmit to the SerDes is clocked into the IOB flip-flops with the reference clock. The GTX_CLK dispatched in the Virtex-II device has nearly the same delay for all IOBs. And the clock to out time of all IOB flip-flops is everywhere the same. Therefore, data to the TLK will be available at the FPGA pins at approximately 90 degrees before the rising edge of the GTX_CLK clock for the SerDes. See Figure 14. This is described in the **Transmitter Interface** and in the TLK250x component series data sheet which is located at: **http://www-s.ti.com/sc/ds/slk2701.pdf**

Control

The TLK control pins, TX_EN and TX_ER, set in the control write register will be re-clocked into flip-flops in the transmitter section. In this way, it is assured that data and control signals are clocked out of the FPGA at exactly the same time.

Pull down resistors have been placed on both control signals in the IOBs with a UCF directive. So when the SerDes is coming out of power down mode, it starts sending IDLE characters.

See Figure 15 for the Transmitter block diagram.

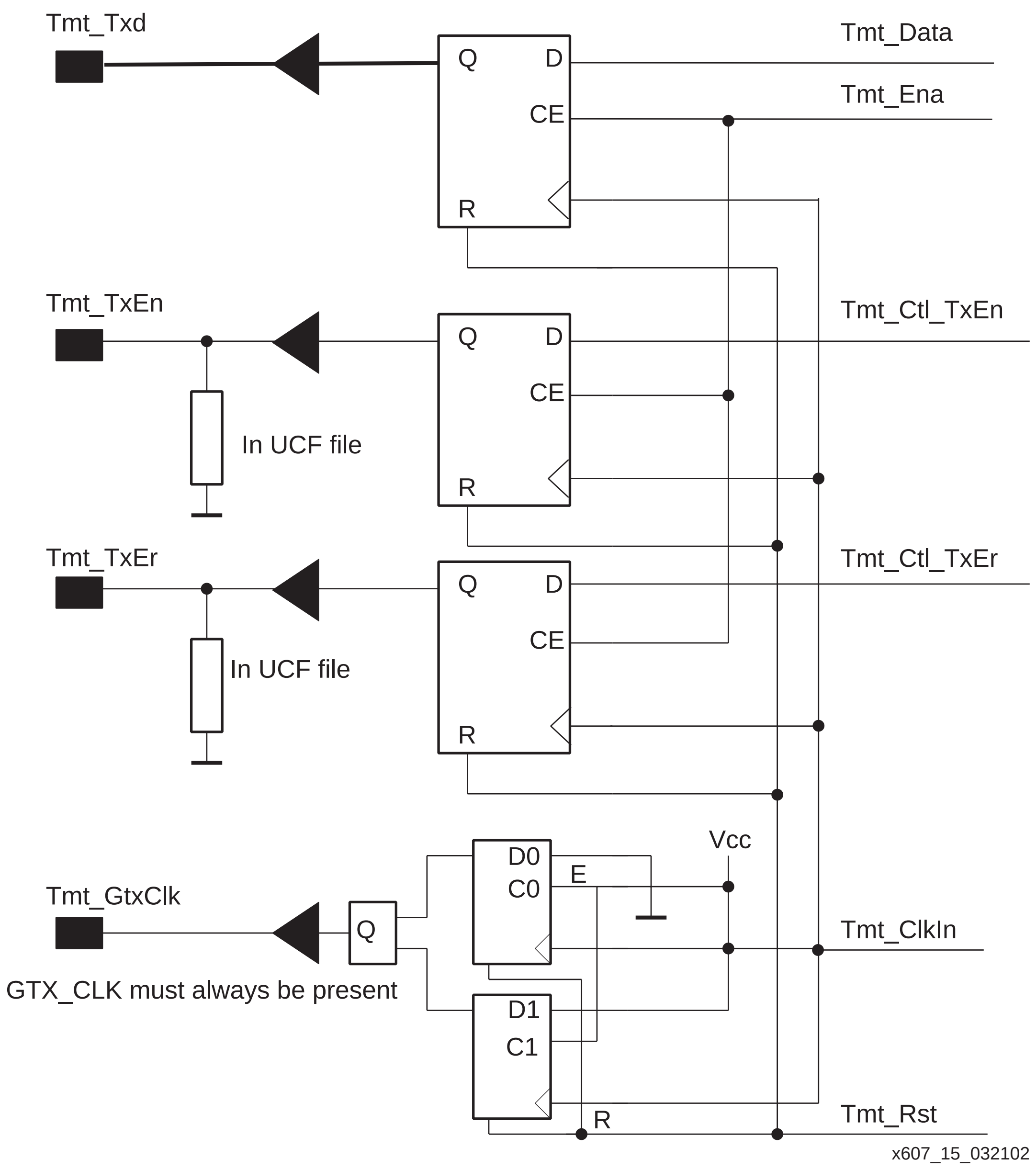

Figure 15: **Transmitter Block Diagram**

Receiver Section

Figure 16 is a diagram of the Receiver register.

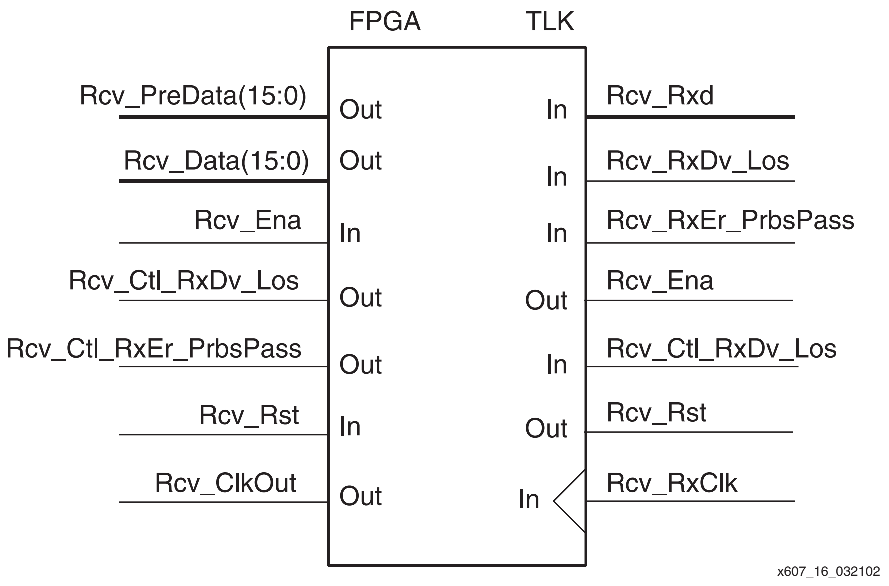

Figure 16: **Receiver Register**

Clock

The SerDes device generates a recovered word (16-bit) clock at the RX_CLK terminal. Clock and received data minimum setup and hold times. See Figure 4.

The data at the RXD[15:0], RX_DV and RX_ER pins will be minimal 3 ns available before the RX_CLK clock, and will last minimal 3 ns after the rising edge of the RX_CLK clock. Knowing this, it is clear that a DCM is needed to distribute the recovered word clock in the FPGA.

Data and Control

Data and status (control) values are clocked into IOB FFs at the rising edge of the RX_CLK clock. Data and status values have a pad to FF input requirement. Therefore, the DCM need to get a slight phase shift for the RX_CLK. The receiver registers are always enabled. This means that every value appearing at the RXD pins will be clocked in at the rising edge of the RX_CLK clock.

Two sets of pipeline registers have been used in the receiver data path. The first pipeline register is clocked on the falling edge of the RX_CLK clock. This increases the possibility to use the received data as decision logic, together with the two control line inputs. Care must be taken that the delay between the two registers is not greater than ~2.2 ns
(2.2 ns + 0.7 ns = ~3 ns).

The second pipeline register (can be a BRAM input, or SRL input as well) is clocked on the rising edge of the RX_CLK. A timing constraint must be applied to make sure that the delay between the two registers is not longer the ~3 ns. The register is enabled by the AND function of Receiver Enable, RX_DV and RX_ER.

This AND function is made with one LUT, and its output is routed to 16 enable inputs of Configurable Logic Block (CLB) FFs. More than 3 ns is needed to enable the FFs, due to the length of the net. Together with the delay between the input registers in the IOB and the first pipeline register there is ~6 ns. This provides enough time to enable the FFs before the data is clocked in. Erroneous data from the input (RxD) is decoded for the control register, but is never passed to the back-end design.

When valid data is presented, RX_ER = 0 and RX_DV = 1, it is always clocked into a register, BRAM, or SRL of the back-end design. The two receiver status bits are clocked into the Control

Read Register at the rising edge of the GTX_CLK clock. Both clocks have the same frequency but are not synchronous with each other. See Figure 17 and Figure 18.

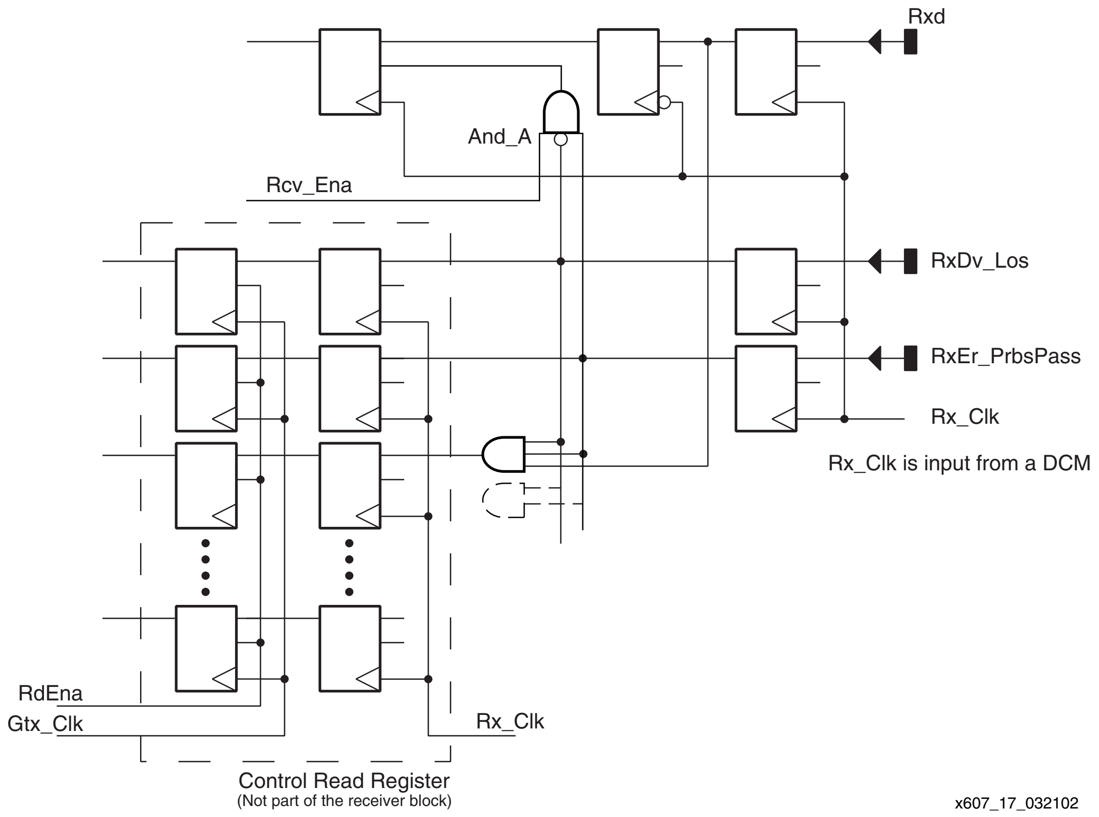

Figure 17: **Receiver Block Diagram**

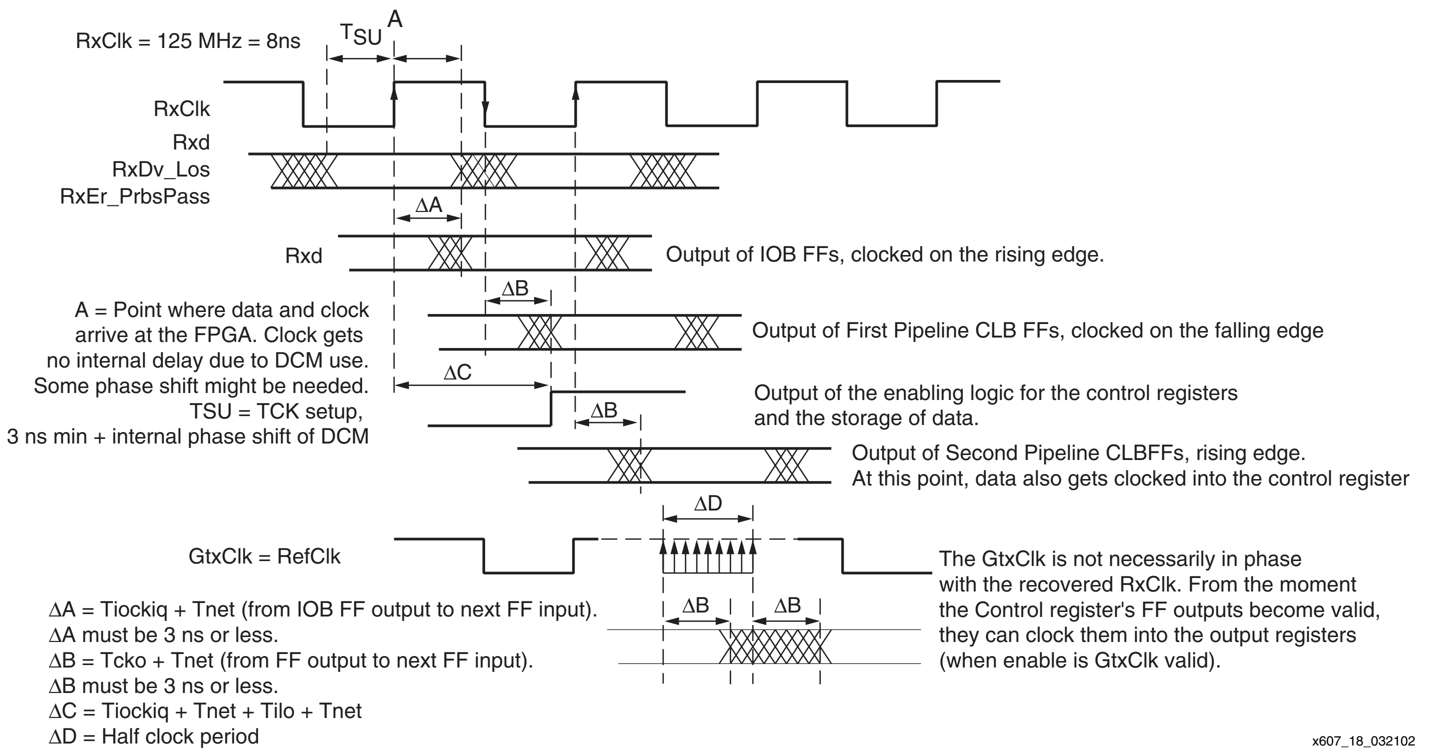
RxClk = 125 MHz = 8ns
TSU
A
RxClk
Rxd
RxDv_Los
RxEr_PrbsPass
ΔA
Rxd
Output of IOB FFs, clocked on the rising edge.
ΔB
A = Point where data and clock arrive at the FPGA. Clock gets no internal delay due to DCM use. Some phase shift might be needed. TSU = TCK setup, 3 ns min + internal phase shift of DCM
Output of First Pipeline CLB FFs, clocked on the falling edge
ΔC
Output of the enabling logic for the control registers and the storage of data.
ΔB
Output of Second Pipeline CLBFFs, rising edge. At this point, data also gets clocked into the control register
ΔD
GtxClk = RefClk
The GtxClk is not necessarily in phase with the recovered RxClk. From the moment the Control register's FF outputs become valid, they can clock them into the output registers (when enable is GtxClk valid).
ΔB
ΔB
ΔA = Tiockiq + Tnet (from IOB FF output to next FF input).
ΔA must be 3 ns or less.
ΔB = Tcko + Tnet (from FF output to next FF input).
ΔB must be 3 ns or less.
ΔC = Tiockiq + Tnet + Tilo + Tnet
ΔD = Half clock period
x607_18_032102

Figure 18: **Receiver Timing**

Interface Design

The internal interface to this TLK interface is application specific. In order to deliver or receive data at the speed of the SerDes interface, transmit and receive FIFOs are needed.

A 1024 by 18-bit is the best fitting block RAM configuration. One block RAM is used as receiver FIFO and one as transmitter FIFO. Using the FIFOs makes it possible to transmit and receive burst of 1024 words. If different size, depending the application, is needed use other FIFO depths.

The interface to the SerDes is 16-bit wide and works at 125 MHz. Because the On-Chip Peripheral Bus (OPB) is 32 bits wide, the application can work with 32-bit wide data paths at half-speed of 62.6 MHz.

When the design can deal with a FIFO size of 512, the 32-bit to 16-bit conversion can be performed in the FIFO.

Notes:

1. Remove the last register pipeline register of the receiver, because the first address storage place in the FIFO can take the function of this register.
2. The address decoding section of the OPB interface can be made simple.
3. The data write register is the first address of the transmit FIFO.
4. The data read register is the last address of the receive FIFO.
5. Use one register for the SerDes control:
   a. Upper 8-bit will be readable.
   b. Lower 8-bit will be writable.

Hardware Interface design

The connection of the FPGA to the TLK SerDes is functioning at a speed of 125 MHz; for 3.125 GHz devices, it is 156.25 MHz.

The following are guidelines for the hardware interface:

1. When using a single SerDes, place it as close as possible to the FPGA.
2. When increasing the distance between the two devices, the PCB tracks start acting as transmission lines. Threat them as such!
   a. Using the DCI function of the FPGA I/O can help with the termination of the created transmission line.
   b. Termination at the TLK side might be needed when both devices are placed far apart.
3. Use different FPGA I/O banks for transmit and receive connections.
4. Consider putting the control signals, RX_ER/PRBS_PASS and RX_DV/LOS, in the middle of the receiver data bus at the FPGA side.
   a. This helps the timing when both signals are used to latch the incoming data.
5. Consider putting all the other control signals in between the two data buses at the FPGA I/O side. An example of a possible FPGA pin layout is shown in Figure 19.
6. Match as much as possible the PCB tracks between the two devices.
   a. Easiest way to do this is by physically placing the TLK device with pins 17 to 32 directed towards the FPGA.
   b. When the control pins have been placed in the middle between the two data buses, they can nearly connect as straight links to the SerDes.
   c. Connect the transmit and receive data buses to both right and left sides of the SerDes. They can easily be matched in length.
   d. The control signals can be somewhat shorter than the data signals, then they will arrive earlier at both sides and help with the timing.

An example of how both devices can be coupled is shown in Figure 20.

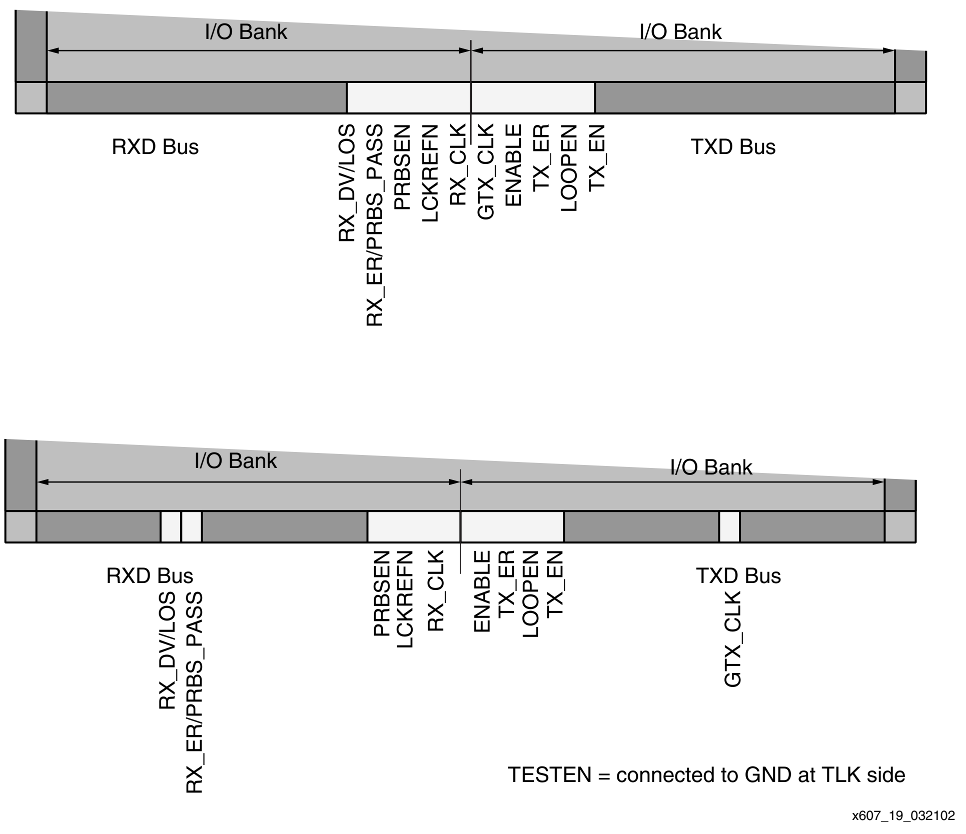

Figure 19: **FPGA to I/O Side**

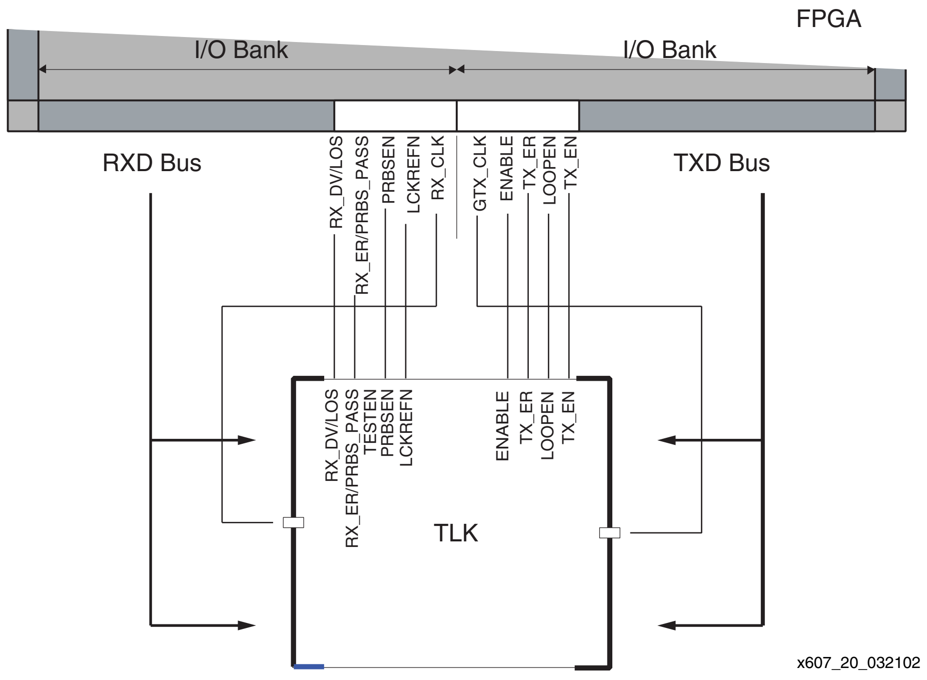

Figure 20: **FPGA to TLK Connection**

Reference Design Files

The reference design files are located can be downloaded from:
ftp://ftp.xilinx.com/pub/applications/xapp/xapp607.zip

Conclusion

This design shows how easy it is to make an interface to a high-speed SerDes device. The strict and narrow timing is the main area to pay close attention to.

Connecting one of these devices to an FPGA consumes many I/O pins (44). Connecting more than one of these devices might require a bigger FPGA, but this can be solved by the following:

- Use packages with multiple (quad) high-speed SerDes devices, or
- Use the Virtex-II Pro™ FPGA device family .

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/17/02	1.0	Initial Xilinx release