XAPP764 (v1.0) May 25, 2004

Connecting Xilinx FPGAs to the Philips A-rate Fibre Optic Transceiver

Summary

This application note shows how a Xilinx Virtex™-II or Virtex-II Pro™ device can connect to a Philips TZA3015HW 30 Mbit/s to 3.2 Gbit/s A-rate™ 4-bit fibre optic transceiver. The reference design with this application note uses the TZA3015HW.

Introduction

The TZA3015HW connects to a framer or other device via a 4-bit, 800 Mbits/s, low voltage differential signaling (LVDS) connection. It has many features and options that can be used across the I^{2}C™ interface.

The reference design uses the source synchronous method to connect the TZA3015HW transceiver to a Virtex-II or Virtex-II Pro device. The "PCB Guidelines" section describes a possible method to connect the TZA3015HW device to a Virtex-II or Virtex-II Pro device.

TZA3015HW

This application note gives only a summary of the TZA3015HW device. For complete functional, software, and hardware descriptions, refer to the Philips TZA3015HW data sheet at:

http://www.semiconductors.philips.com/acrobat/datasheets/TZA3015HW_4.pdf

Key Features

The key features of the TZA3015HW are:

- A-rate interface supports any bit rate from 30 Mbits/s to 3.2 Gbits/s with one single reference frequency when using the I^{2}C interface
- 4-bit parallel interface
- Selectable half-clock-rate Double Data Rate (DDR) or Single Data Rate (SDR) clocking scheme on parallel interface, enabling easy interfacing with FPGA devices
- Eight pre-programmed (pin selectable) bit rates:
  - SDH/SONET rates at 155.52 Mbits/s, 622.08 Mbits/s, 2488.32 Mbits/s, and 2666.06 Mbits/s (STM16/OC48 + FEC)
  - Gigabit Ethernet at 1250 Mbits/s and 3125 Mbits/s
  - Fibre Channel at 1062.5 Mbits/s and 2125 Mbits/s
- ITU-T compliant jitter tolerance for the Device Control Register (DCR)
- ITU-T compliant jitter transfer for DCR in clean-up loop back mode
- ITU-T compliant jitter generation for synthesizer
- I^{2}C-bus and pin programmable (configurable options)
  - Programmable frequency resolution of 10 Hz
  - Independent receive and transmit bit rates
  - Slice level adjustment to improve the bit error rate (BER)
  - Six reference frequency ranges

  - Adjustable swing for Current Mode Logic (CML) serial data and clock outputs
  - Programmable polarity of RF I/Os
  - Clock versus data swap for optimum connectivity
  - Swap of parallel bus for optimum connectivity
  - Mute function for a forced logic 0 output state
  - Programmable parity
  - Programmable 32-bit frame detection
- Six selectable reference frequency ranges
- Transmitter, receiver and transceiver modes
- Clean-up loop back mode
- Line loop back mode
- Diagnostic loop back mode
- Serial loop timing mode
- Single 3.3V power supply

TZA3015HW Block Diagram

Figure 1 shows a block diagram of the TZA3015HW.

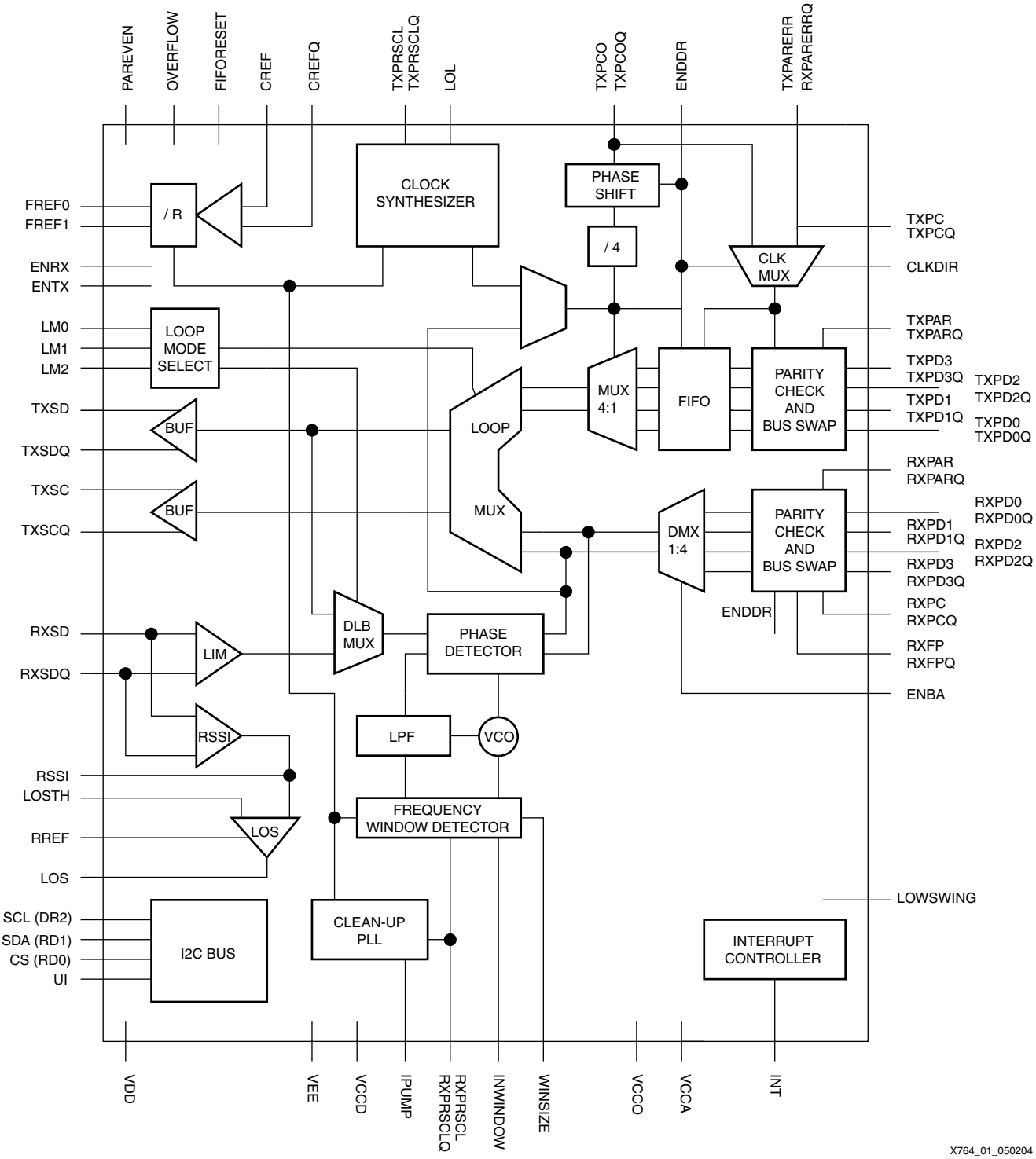

Figure 1: **TZA3015HW Block Diagram**

Functional Description

Only the functions used in the reference design are described in this section. All other functionalities of the optical transceiver are found in the TZA3015HW data sheet.

Configuration

After a power on reset, the TZA3015HW is initialized with a default set of parameters, listed in the TZA3015HW data sheet.

The optical transceiver has two modes, determined by the state of the UI pin (see Table 1):

- I²C-bus
- Eight pre-programmed modes

Table 1: **User Interface Mode Selection through UI Pin**

UI (Pin 13)	Mode	Pin 22	Pin 23	Pin 24
LOW	Pre-programmed	DR0	DR1	DR2
HIGH	I²C-bus	CS	SDA	SCL

Selection of the pre-programmed user-interface mode can be done either through three jumpers, whose combination determines the used protocol, or by connecting the three selection lines to the FPGA with the select register.

This design focuses on the I²C user interface. Figure 2 shows a sample connection circuit, with all interconnect possibilities enabled.

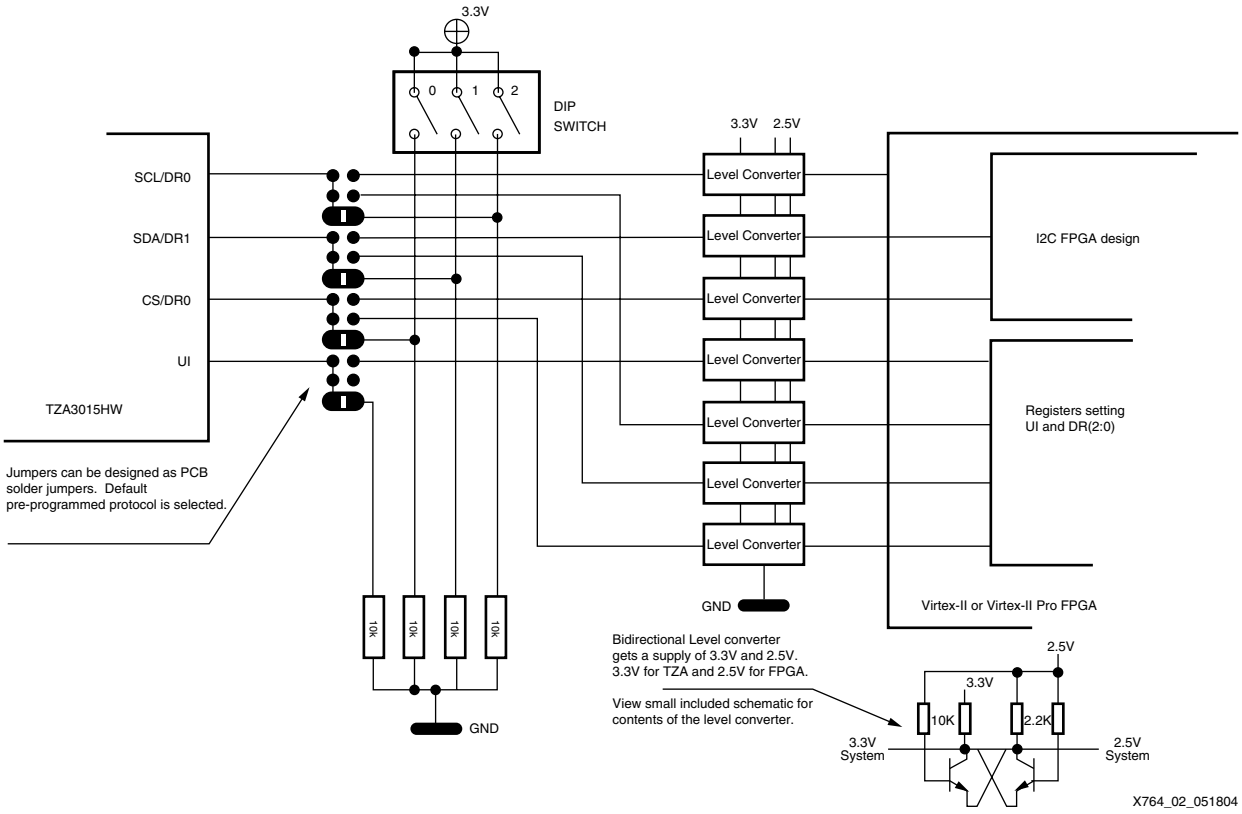

Figure 2: **I²C and/or Preprogrammed Mode Selection**

Note: When using 3.3V I/O for the Virtex-II Pro devices, the level conversion is not needed.

Reference Clock

The reference clock, connected to pins CREF (pin 93) and CREFQ (Pin 94), is used for both the DCR frequency window detector and the transmitter synthesizer. Table 2 shows the encoding of the FREFI2C bits in the I²C mode register.

Table 2: **Reference Frequency Settings through I²C Initialization**

Bit			Division Factor R	Reference Frequency Range (MHz)
FREFI2C2	FREFI2C1	FREFI2C0		
0	0	0	1	18 to 21
0	0	1	2	36 to 42
0	1	0	4	72 to 84
0	1	1	8	144 to 168
1	0	0	16	288 to 336
1	0	1	32	576 to 672

Parallel Bus Clocking Schemes

The TZA3015HW supports both co-directional and contra-directional clocking schemes for the parallel data bus. In I²C mode, the clocking scheme is set through the CLKDIR bit of the MUXCON0 register at address F1h (see Table 3).

Table 3: **I²C Interface Connection Type Settings**

CLKDIR Bit	Clocking Scheme
0	Contra-directional
1	Co-directional

Figure 3 shows co-directional clocking (source synchronous clocking). The parallel data clock is generated in the data processing device. The transceiver accepts the data at the TXPC(Q) (pins 64 and pin 65) clock rate.

The reference clock of the transceiver provides the system clock for the FPGA via TXPCO (pin 79) and TXPCOQ (pin 80).

Two advantages to co-directional clocking are:

- When the TZA changes clock frequency, the FPGA design can adapt automatically (after a DCM reset) to a changed TZA frequency.
- The FPGA does not need an extra oscillator.

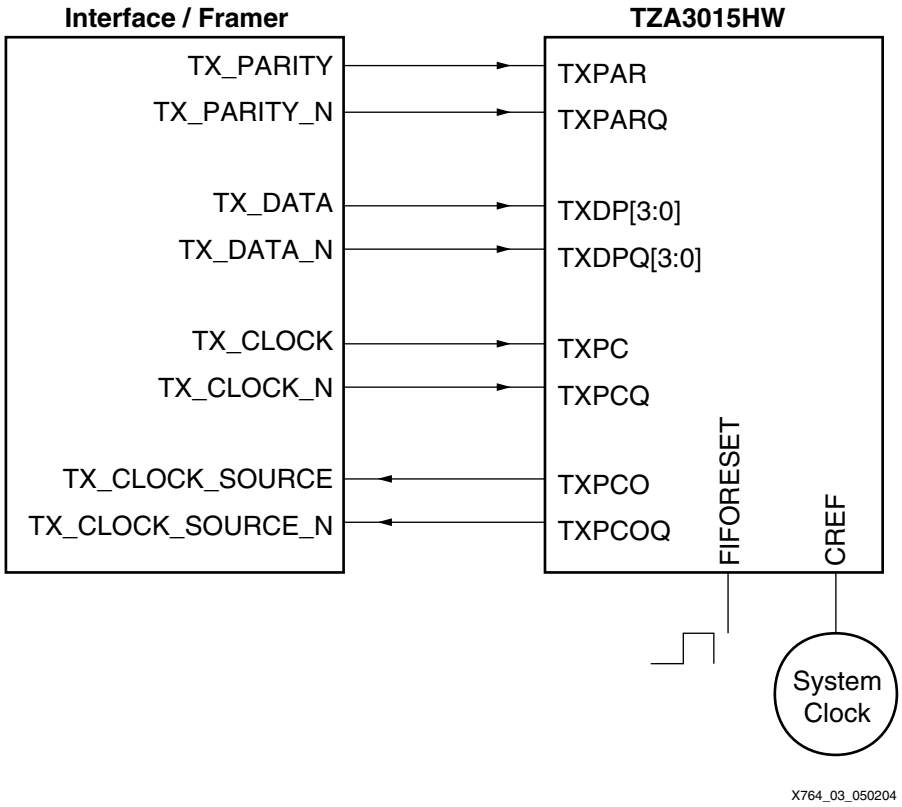

Figure 3: **Source Synchronous Interface**

The TXPCO and TXPCOQ clocks can be shifted over 90 degrees with respect to the internal transceiver clock. In I^{2}C mode, the TXPCOINV register bits of the TXMFOUTC register (F2h) and the TXPOPHASE bit of the MUXCON0 register (F1h) are set as indicated in Table 4. This clock phase shifting cannot be done in the Pre-programmed mode.

Table 4: **Clock Phase Shift Solutions (I^{2}C Only Solution)**

TXPCOINV	TXPOPHASE	PHASE SHIFT
0	0	0
0	1	90
1	0	180
1	1	270

Double Data Rate Mode

The default setting for the parallel clock frequencies TXPC, RXPC, and TXPCO equals the parallel data bit rate. For example, a parallel bit rate of 800 Mbit/s equals a 3.2 Gbit/s serial bit rate. The data is clocked with a 800 MHz clock.

The Xilinx interface requires the use of a half data bit rate clock, called DDR mode operation. This DDR functionality is enabled via the I^{2}C mode by setting bits in the DDR&RXPRSCL (D5h), MUXCON0 (F1h), and TXMFOUTC (F2h) registers. DDR control can only be obtained when the I2CDDR bit of the DDR&RXPRSCL register is set. Table 5 lists the encoding for DDR mode.

Table 5: **Selection of DDR Mode (I^2C Solution)**

Bit	Value	Mode
DDR&RXPRSCL Register		
RXPCDDREN	1	RXPC in DDR mode
	0	RXPC default
MUXCON0 Register		
TXPCDDREN	1	TXPC in DDR mode
	0	TXPC default
TXMFOUTC Register		
TXPCODDREN	1	TXPCO in DDR mode
	0	TXPCO default

I/O Configuration

This section describes the register bits that configure the I/O.

LVDS Output Level

The LOWSWING bit of the MFOBON register (A4h) sets the output swing of all LVDS outputs. This bit is enabled when the I2CLOWSWING bit in the MFOBCON register is set.

Parallel Clock Output Polarity

The RXPCINV bit of the RXMFOUTC0 register (D4h) sets the polarity of the RXPC(Q) output clock. The RXPCEN bit of the same register enables or disables the RXPC(Q) clock.

Parallel Data Output Bus Swap

The RXBUSSWAP bit of the DMXCON register (B8h) is used to swap the bits of the output data bus (RXPD0(Q) – RXPD3(Q)). The DMXMUTE bit of the DMXCON register forces the bus to an all zero state (mute).

The RXPDINV bit of the RXMFOUTC0 register determines the polarity of the parallel data outputs (RXPD0(Q) – RXPD3(Q)). The RXPDEN bit of the RXMFOUTC0 register enables or disables the parallel output data bus.

Frame Pulse Output

The RXPARINV bit of the RXMFOUTC0 register sets the polarity of the RXPAR(Q) output. The RXPAREN bit of the RXMFOUTC0 register enables or disables the frame or parity output.

The TXPARERRINV bit of the TXMFOUTC register (F2h) sets the parity of the parity error output TXPARERR(Q). This output can be enabled and disabled by the TXPARERREN bit of the TXMFOUTC register.

Parallel Data Input Bus Swap

The TXBUSSWAP bit of the MUXCON register (F0h) enables a bus bit order swap of the parallel input data bus.

CMOS Inputs and Outputs

Table 6 summarizes the CMOS inputs and outputs of the TZA3015HW optical device.

Table 6: **Single-Ended, CMOS I/O Component Pin Descriptions**

Pin	In/Out	Function
CLKDIR	Input	Co- or Contra-directional
DR0:DR2	Input	Data rate selection
ENBA	Input	Enable byte alignment
ENRX	Input	Enable receiver input
ENTX	Input	Enable transmitter input
ENTXSC	Input	Enable serial clock
FIFORESET	Input	FIFO reset
FREF0, FREF1	Input	Reference frequency selection
INT	Output	Interrupt (open drain)
INWINDOW	Output	Frequency window detector
LM0:LM2	Input	Loop mode selection
LOL	Output	Loss of lock
LOS	Output	Loss of signal
LOWSWING	Input	LVDS voltage swing
OVERFLOW	Output	FIFO overflow alarm
PAREVEN	Input	Odd or even parity check
UI	Input	User interface select
WINSIZE	Input	Frequency detect window

For fixed transceiver settings, control input pins can be tied to GND or VCC on the PCB. Partial flexibility is obtained when inputs are routed to sets of jumpers, and full flexibility is achieved when the control input pins are connected to the FPGA. The TZA control lines are now under control of the FPGA hardware or the internal processor.

DDR Low Voltage Differential Signaling (LVDS)

This section presents the principles targeted towards the Philips A-rate Fibre Optic Transceiver device. The DDR LVDS interface is described in XAPP265 (see "Reference Material").

Data transmit and receive interfaces use four differential data pairs. The transmit interface generates the clock for the transmit data towards the TZA transceiver. The FPGA receiver interface accepts data and a receive clock from the TZA. The FPGA system clock is also provided by the TZA3015HW. This clock, TXPCO(Q), is used in DDR mode and runs at 400 MHz.

Transmitter

A basic, one-bit DDR LVDS transmitter block is shown in Figure 4. The Philips transceiver needs four differential data bits.

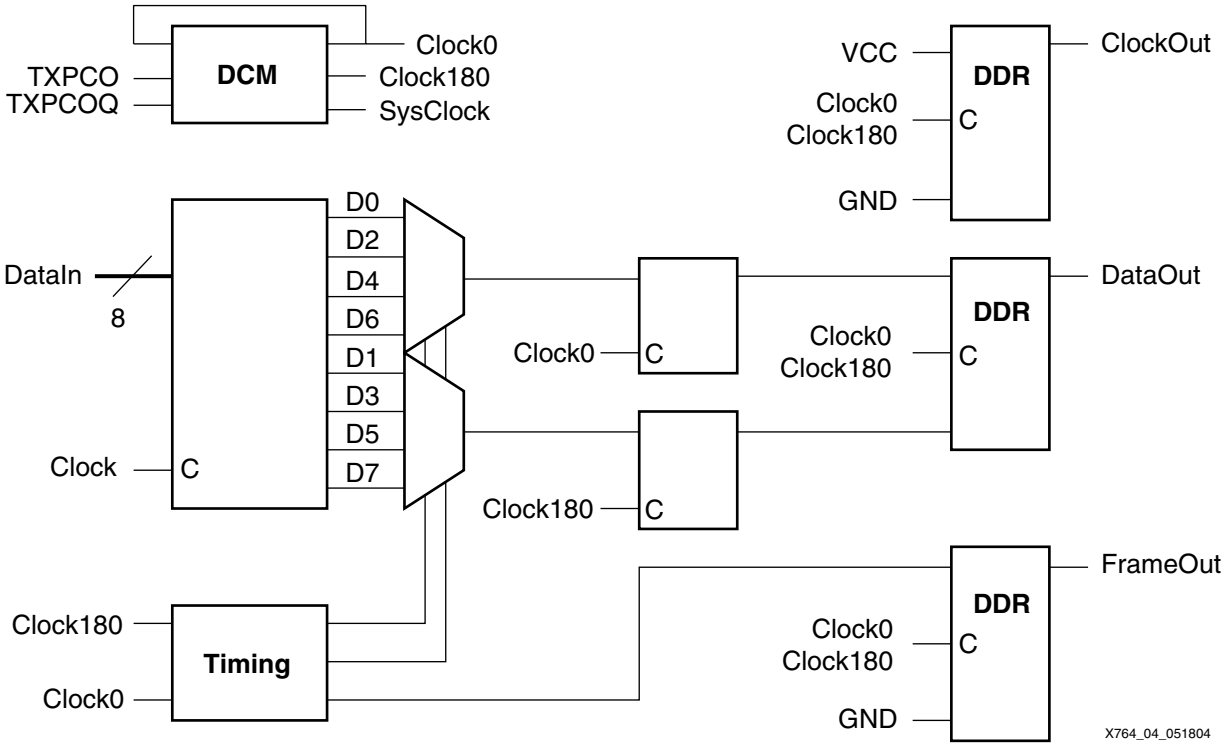

Figure 4: **One-bit Transmitter Interface**

Figure 4 shows a basic, one-bit DDR LVDS transmitter block. The Philips transceiver needs four of these blocks (except for the DCM).

The DCM is fed with a high-speed clock from the TZA device, TXPCO(Q). This high-speed clock is divided by 4 to generate an FPGA system clock of 100 MHz. The phase-aligned clock_0 and its opposite, clock_180, are used for low-jitter operation of the DDR output flip-flops.

Eight data bits registered at the FPGA system clock are split into an even and an odd nibble using the positive edges of clock_0 and clock_180. The two nibbles are then serialized in the IOB using the DDR capabilities.

Figure 5 shows the timing of this process.

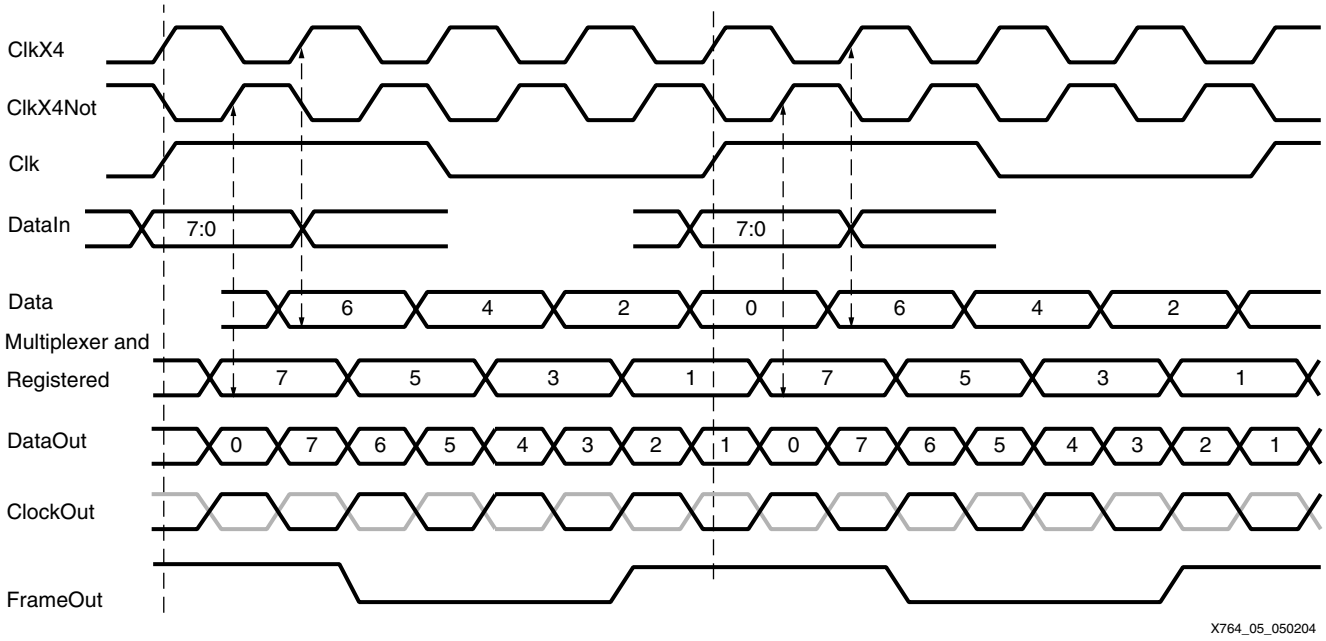

Figure 5: **Transmitter Timing Diagram**

This reference design contains a 4-bit DDR transmitter module written in HDL. The module is assembled as a set of RLOCs (Relational Placed Macros), making it easy to position the transmitter interface inside the FPGA.

UCF and NCF files are available for placing the transmitter module at both sides in I/O banks 2, 3, 6, and 7 of the FPGA.

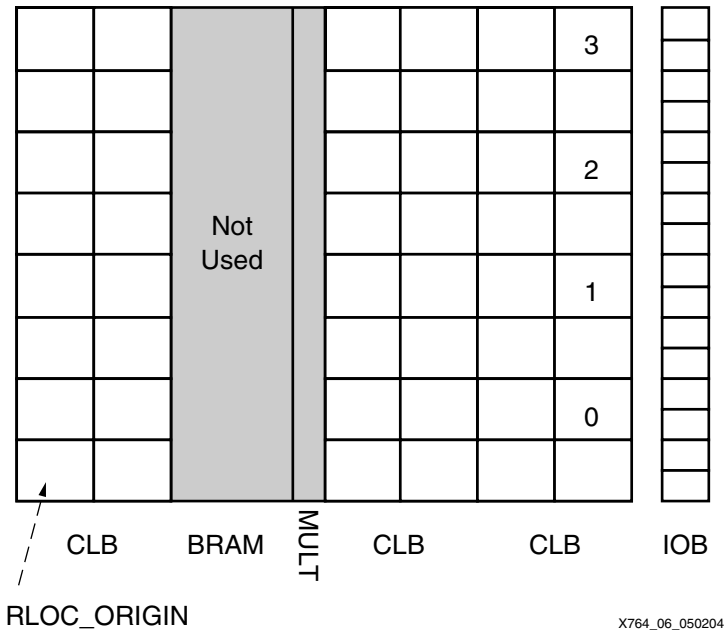

Figure 6: **Transmitter Placement and Layout on the Right Side (Bank 2 and/or 3)**

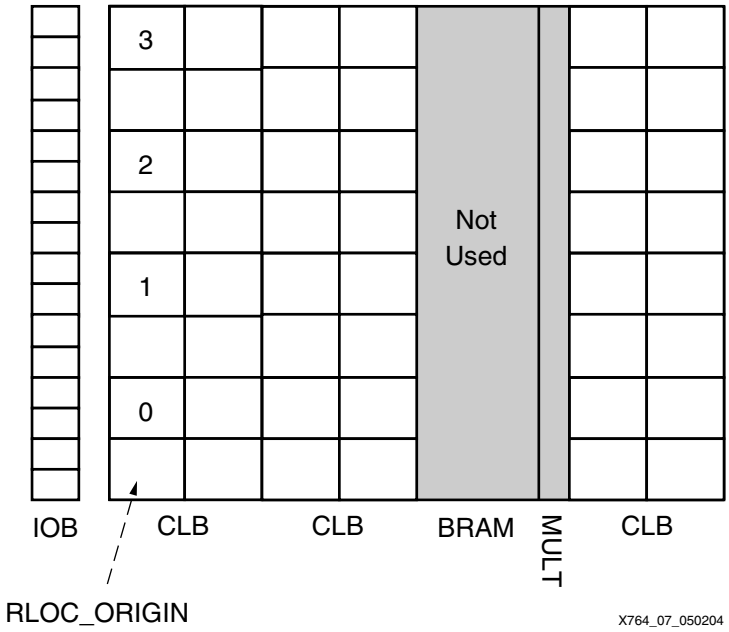

Figure 7: **Transmitter Placement and Layout on the Left Side (Bank 6 and/or 7)**

As shown in Figure 6 and Figure 7, the macro module is four CLBs high and three CLBs wide in order to fit the pitch of the Block SelectRAM™ memory. This configuration is very useful when a FIFO is implemented as part of the transmitter macro, because it immediately connects to the transmitter module with the shortest possible connections.

Figure 8 shows the bit pattern transmitted by a 4-bit module with 32-bit parallel input.

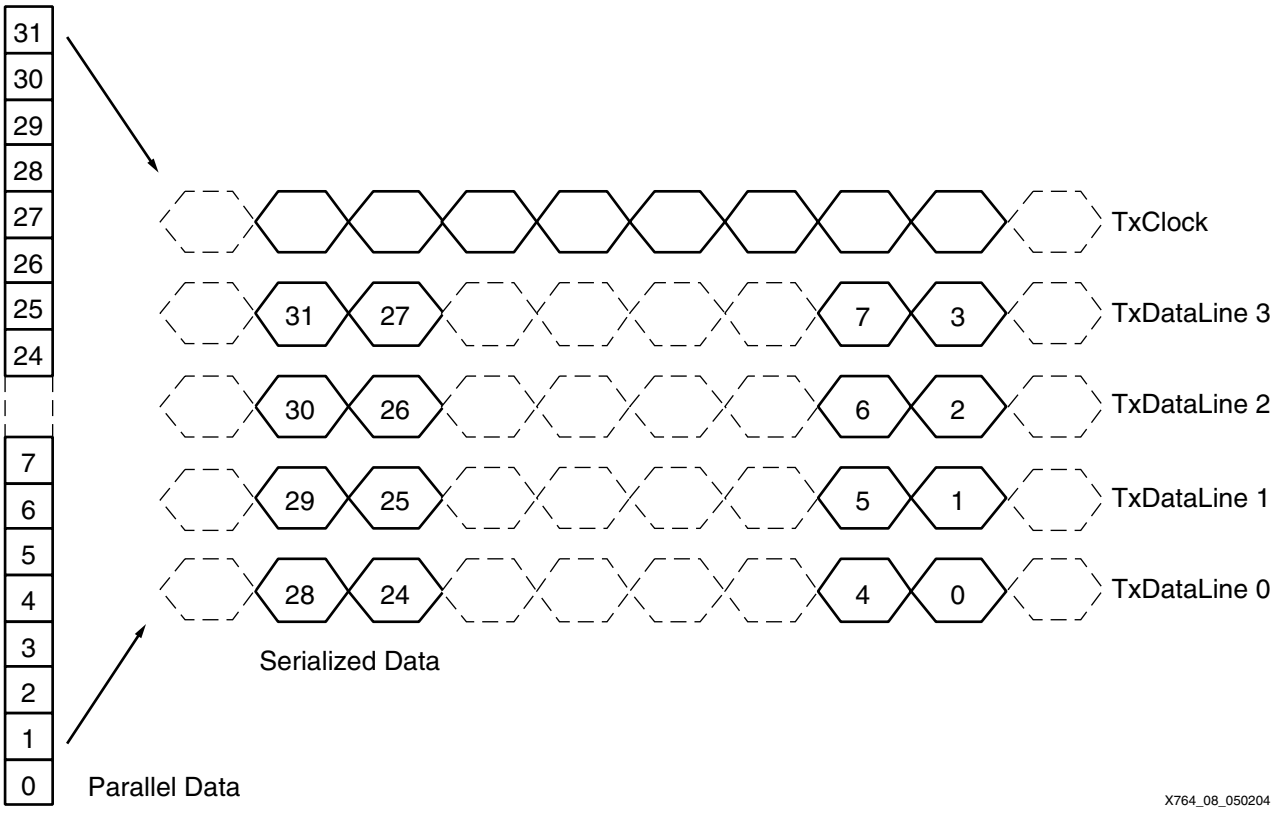

Figure 8: **A 4-bit Transmission Order**

If a different bit pattern is needed during transmission and reception, connect the parallel-to-serial and serial-to-parallel interfaces differently in the transmitter and receiver modules (RPMs).

Receiver

The TZA transceiver transmits edge-aligned data and clock. Assuming that the PCB layout takes this into account, data and clock will arrive aligned at the FPGA pins.

To register the received data into the FPGA with the received clock, the DCM is used in phase shift mode. Fixed phase shifting can be used, dynamic phase shifting is a better choice as indicated in the "Auto Phase Shift DCM" section.

Figure 9 shows a one-bit receiver module. This module takes in one-bit serial differential data and outputs 8-bit parallel data. For the Philips transceiver, four of these modules are assembled together in one basic building block, giving 4-bit differential data in and 32-bit parallel data out to the FPGA.

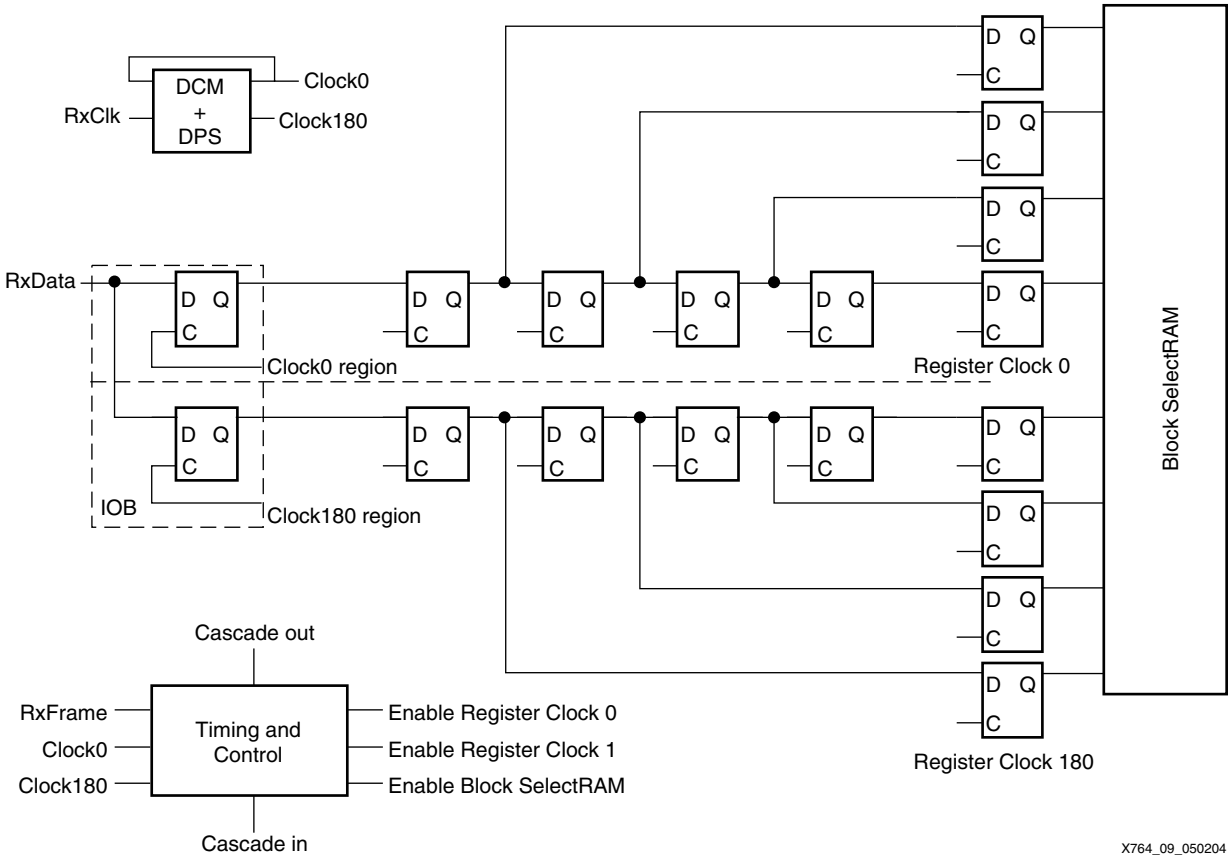

Figure 9: **One-bit Receiver Block Diagram**

In the one-bit design, the received clock is fed into a DCM. This DCM applies a phase shift to the received clock, so that the presented data is registered in the FPGA at the right moment, in the middle of the data 'eye'. This phase shift can be fixed or dynamic, as used in the application example.

Data is presented in the DDR format, meaning that it must be clocked on the rising and falling edges of the clock. The outputs of the DCM are a phase-shifted clock and a 180-degree phase-shifted clock. Both clocks can be used on the rising edge.

The eight serial incoming data bits are split into two nibbles, one clocked on the corrected (phase-shifted) clock and one clocked on the 180-degree shifted clock. The result is an 8-bit parallel word with a jumbled data bit order. Correctly wiring the data bits to the receiver FIFO resolves the ordering.

The receiver FIFO, which is the next stage in the receiver module, offers an easy interface. It is constructed with Block SelectRAM memory and uses the self-addressing principle, as described in XAPP291 (see "Reference Material"). Refer to the "Self-Addressing FIFO with Depth Extension" section for a way to extend the depth of the FIFO.

As with the transmitter, this receiver module is built as a macro containing RPM directives. The reference design provides UCF and NCF files and additional information to allow placement at both sides (I/O banks 2, 3, 6, and 7) of the FPGA 'die'. Figure 10 shows the module setup for the receiver at one side of the FPGA.

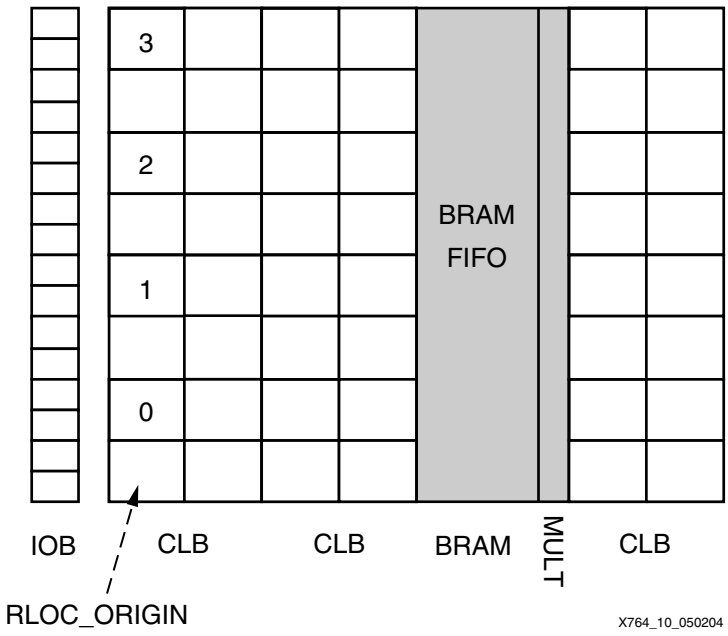

Figure 10: **4-bit Receiver Placement and Layout for Bank 6 and/or 7**

Auto Phase Shift DCM

The Digital Clock Manager (DCM) phase shift capability is used to position the receiver clock exactly in the middle of received data bit width, in order to let the receiver circuit work in all circumstances. This principle has also been described in XAPP268 (see "Reference Material").

The clock and data are aligned when the TZA device is transmitting them towards the FPGA interface. It is asked to perform the PCB layout so that data and clock traces have the same length. Due to different layout and circuit conditions there can be variations in length between the different PCB traces.

A certain amount of clock to data length variation can be absorbed by the DCM when using its phase shift capabilities.

In the Virtex-II and Virtex-II Pro device families, all Input/Output Blocks (IOB) are the same, except those used as clock inputs. These IOB have an extra direct input to the DCM clock input. When these IOBs are used as clock inputs, the normal IOB logic is omitted.

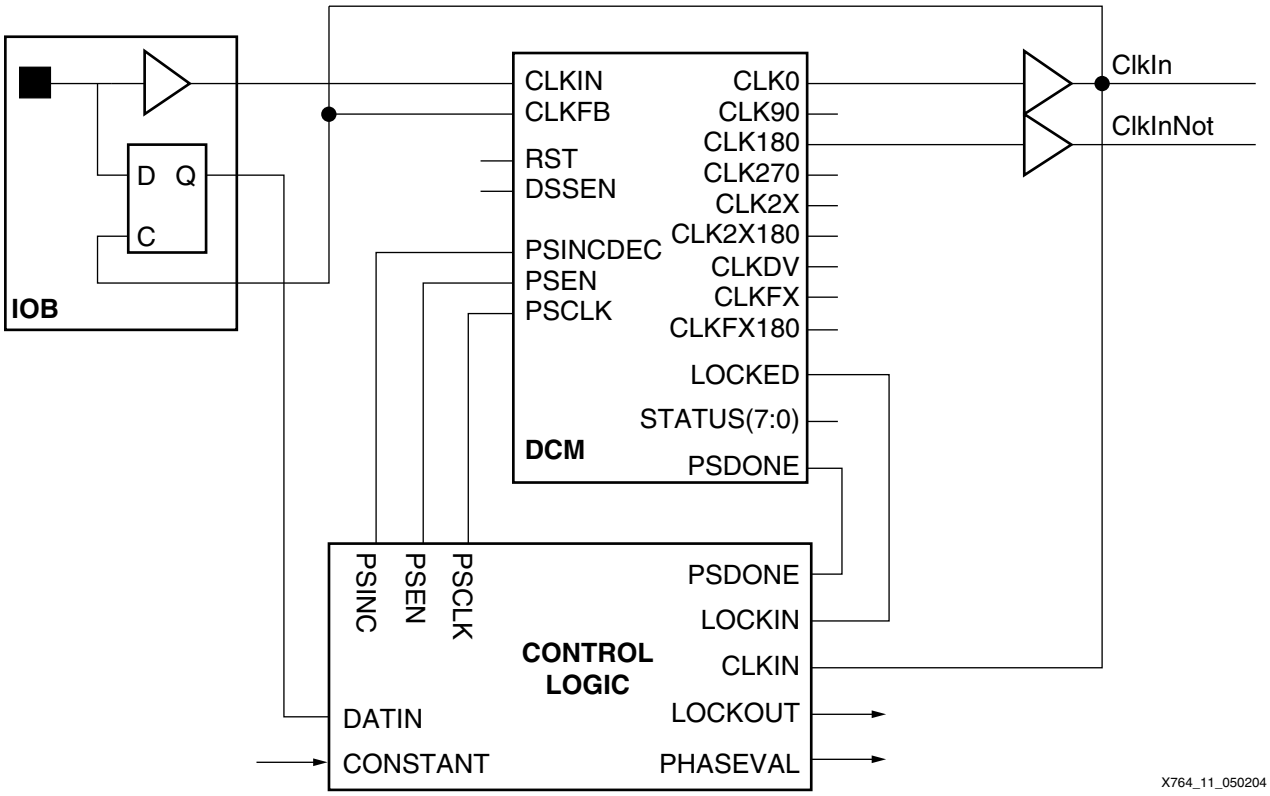

Figure 11: **Auto Phase Shift DCM Setup**

In the circuit shown in Figure 11, the incoming clock is used as the clock input and as the data input for the IOB FF. The DCM-adjusted input clock registers itself at the IOB flip-flop.

The DCM is set in variable mode with an initial phase shift of –255. The equivalent counter in the control logic is set to zero, as is the internal state machine. When taken out of reset, the DCM adjusts its output clocks and indicates this process with the “locked” output.

Now the DCM is incremented by one, as shown in Figure 12, from the control logic. The control logic counter is incremented by the PSDONE signal from the DCM. When this operation happens, the IOB flip-flop is checked. When it is "0", the DCM is again incremented. When changed, the current count value is stored (ps0) and the state machine takes state one. Incrementing continues until the input flip-flop changes back to "0". The count value at this point is also stored (ps1) and the value ps3 (= ps1 – ps0) is calculated. The DCM is now decremented until its value is ps3.

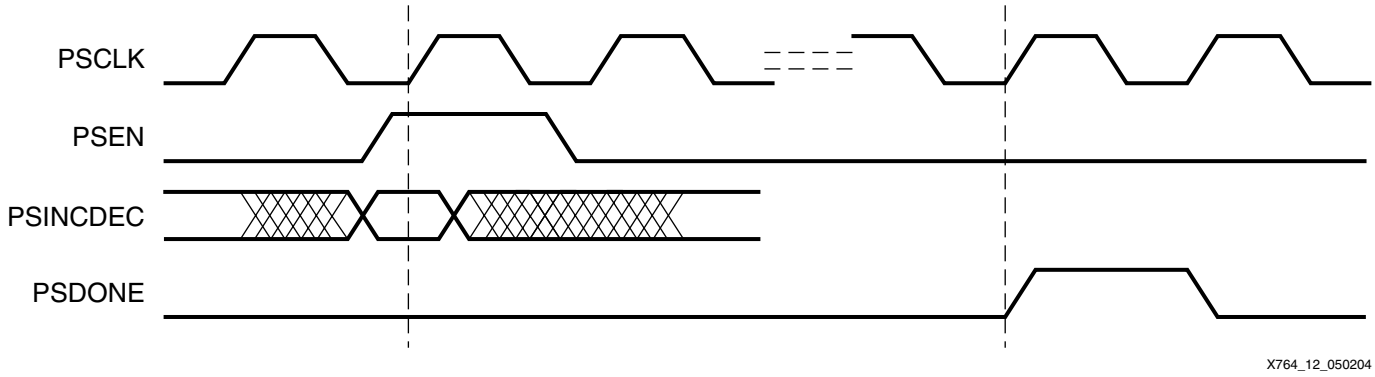

Figure 12: **Phase-Shift DCM Interface Timing**

Setup is finished, and the LOCKOUT output of the control logic is driven High, which performs the same function as the LOCKED output of the DCM, indicating the startup of the DCM is finished.

Self-Addressing FIFO with Depth Extension

FIFOs are constructed from memory, read counters (pop), write counters (push), and flagging logic. With a self-addressing FIFO, the counters are replaced by the memory itself. Thus clock skew is no longer an issue.

Figure 13 shows a standard self-addressing FIFO. Part of the memory data output fed back as memory address. At the same time, that part is also incremented with some logic and injected in the memory with the incoming data.

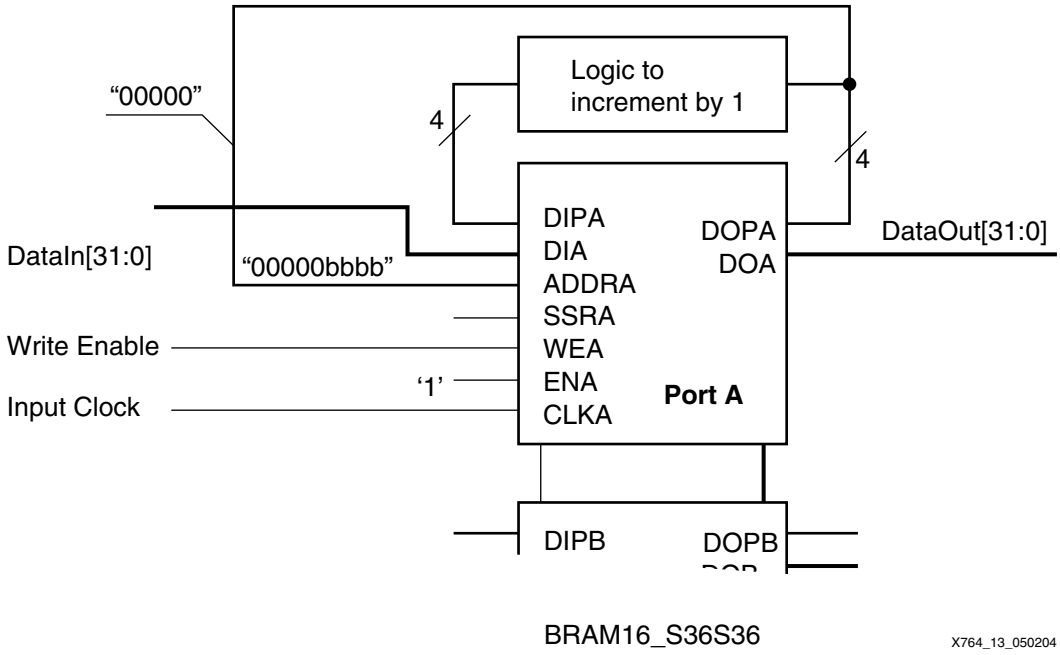

Figure 13: **Standard Self-Addressing FIFO**

At the start of operation, the output of the memory is assumed to be all zeros. The RAM is addressed with this value from the feedback part of the output. At the same time, the data is incremented (in the adder) by one, and the result is presented at the memory data input together with the incoming data.

The next clock edge addresses memory space "zero" and writes the data value "zero+1 & input data". Due to the construction of the memory blocks, this value appears at the output after a "Clock-to-output (Tbcko)" time.

Now memory space "one" is addressed and a data value of two (output one + added one) is presented.

The address increment logic is built from a LUT and does not require any clock. If a clock is needed, the advantage of direct data out is lost.

Block SelectRAM memories have the ideal design for this mechanism. When a 32-bit wide FIFO is needed, the extra four bits (parity bits) can be used for the address storage-incrementer functionality.

With four bits, a 16-deep FIFO can be constructed. If a deeper FIFO is needed, the following solutions are possible:

- Use normal data bits

  This solution limits the width of the input data. For a 32-bit requirement, one Block SelectRAM memory is not enough.

- Use an extra adder

  This solution extends the address with the extra adder. When the address reaches the maximum value available in memory, one is added to the result. The extra adder does not need to run at full FIFO speed, but the increment to the BRAM timing must be controlled. Figure 14 shows an address-extended self-addressing FIFO.

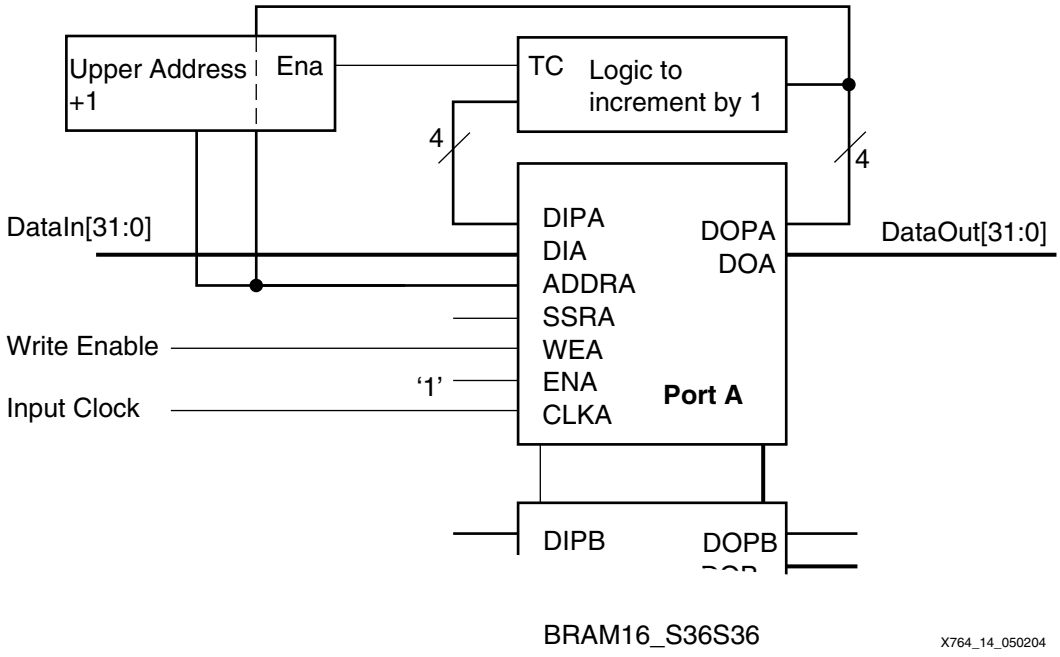

Figure 14: **Address-Extended Self-Addressing FIFO**

A sample design is included in the reference design.

I2C Interface

This section describes the protocol of the I2C-bus and the design available in the FPGA. The I2C-bus is only used in Master mode and is used to initialize the TZA components.

For complete information on the I2C protocol, refer to the I2C-bus specification at:

http://www.semiconductors.philips.com/acrobat/literature/9398/39340011.pdf

The I2C-bus is a two-wire communication bus, containing serial data (SDA) and serial clock (SCL). These signals carry information between devices connected to the bus. The number of devices connected to the same bus is limited only by a maximum bus capacitance of 400 pF. The SDA and SCL lines are bidirectional, connected to a positive supply voltage via a pull-up resistor.

Communication on the I2C-bus is only possible when the bus is not busy, meaning SDA and SCL are High. The output stages of devices connected to the bus must have an open-drain or open-collector in order to perform the needed wired-AND functionality.

Normally, each device on the bus has a unique address and can operate as either a transmitter or receiver. Devices can also function as Masters or Slaves. A Master device initiates a data transfer on the bus and generates the clock signals to permit that transfer. Any other device

being addressed is considered to be a Slave. The TZA3015HW always acts like a Slave device, and all TZA transceivers use the same address. Therefore, the FPGA I²C interface is configured as a Master, and a CS (Chip Select) pin selects the different optical TZA transceivers.

The I²C protocol defines an arbitration procedure ensuring that if more than one Master simultaneously tries to control the bus, only one is allowed to do so, which keeps the message from being corrupted. The reference design supports the arbitration and clock synchronization procedures defined in the I²C specification.

Start and Stop Conditions

Data transfers are initiated with a START condition and are terminated with a STOP condition.

When the I²C-bus is not busy, both clock and data lines are High.

The START condition is defined by a High-to-Low transition on SDA while SCL is High. Likewise, the STOP condition is defined by a Low-to-High transition on SDA while SCL is High. The definitions of data, START, and STOP ensure that the START and STOP conditions are never confused as data (see Figure 15).

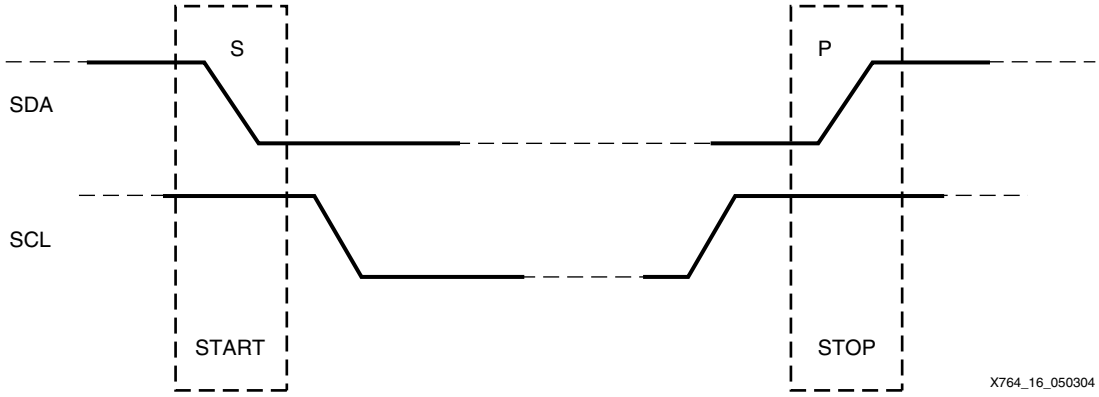

Figure 15: **I²C Waveforms**

Acknowledge

Each data packet on the I²C-bus consists of eight bits of data followed by an acknowledge bit. One complete data byte transfer requires nine clock pulses. The most significant bit (MSB) of data is transferred first. The transmitter releases SDA during the acknowledge, and the receiver of the data transfer must drive SDA Low during the acknowledge to acknowledge correct receipt of data.

If SDA is not pulled Low during the acknowledge bit by a Slave receiver, this case indicates that the Slave receiver was unable to accept the data. The Master can then generate a STOP condition to abort the transfer.

The following operation applies to general I²C cases, but not to TZA devices. If the Master receiver does not generate an acknowledge, a transmitter interprets this condition as the last byte of a data transfer. The transmitter must leave SDA High to enable the Master to generate a STOP condition.

Read/Write Protocol

Standard communication on the bus between a master and a slave is composed of four parts:

(S)tart, slave address, data transfer, and sto(P).

After the START condition, a slave address is sent. This seven-bit address is followed by an eighth bit, which is the read/write (R/W) bit. A "1" on the R/W bit indicates a request for data (read) and a "0" indicates a data transmission (write). See Figure 16.

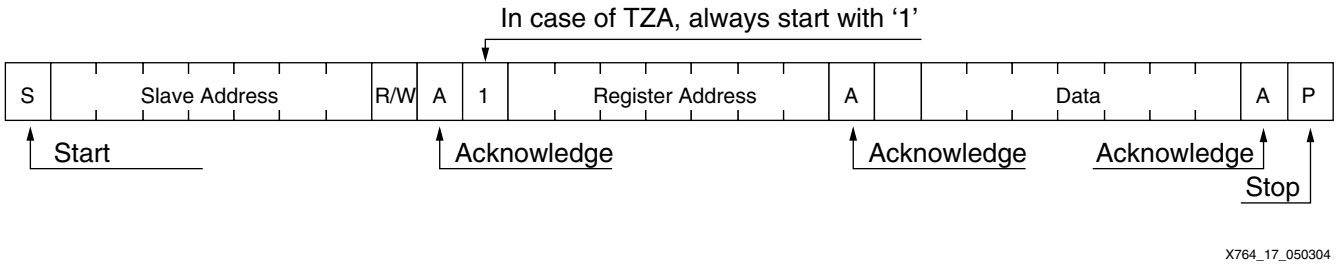

Figure 16: **I2C Master Communications Protocol Setup**

Only the slave with a calling address that matches the address transmitted by the master can respond. It sends an Acknowledge bit (A) back by pulling SDA Low on the ninth clock.

Once slave addressing is achieved, the data transfer can occur byte-by-byte as specified by the R/W bit. When communicating with the TZA optical transceiver, the data transfer has the following form:

Register Address, Data Byte

The Master terminates the communication by generating a STOP condition to free the bus. However, the Master may generate a START signal without generating a STOP signal first. This condition is called a repeated START.

Figure 17 shows the protocol for reading data from a TZA device. The Master device sends the slave address. The CS for the correct TZA device must be set. Data is received after acknowledgment by the Slave. The Master must send an acknowledgment or, when finished, a non-acknowledgment followed by a STOP.

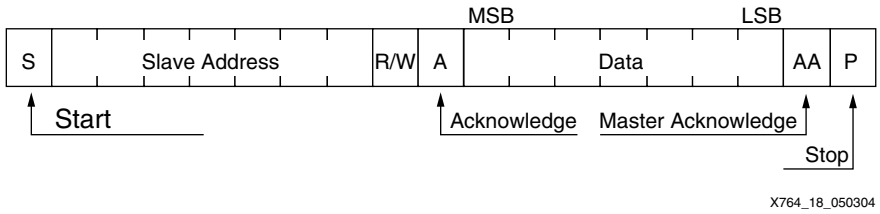

Figure 17: **I2C Slave Answer Back Setup**

I2C Design

Figure 18 shows the block diagram of the included I2C reference design.

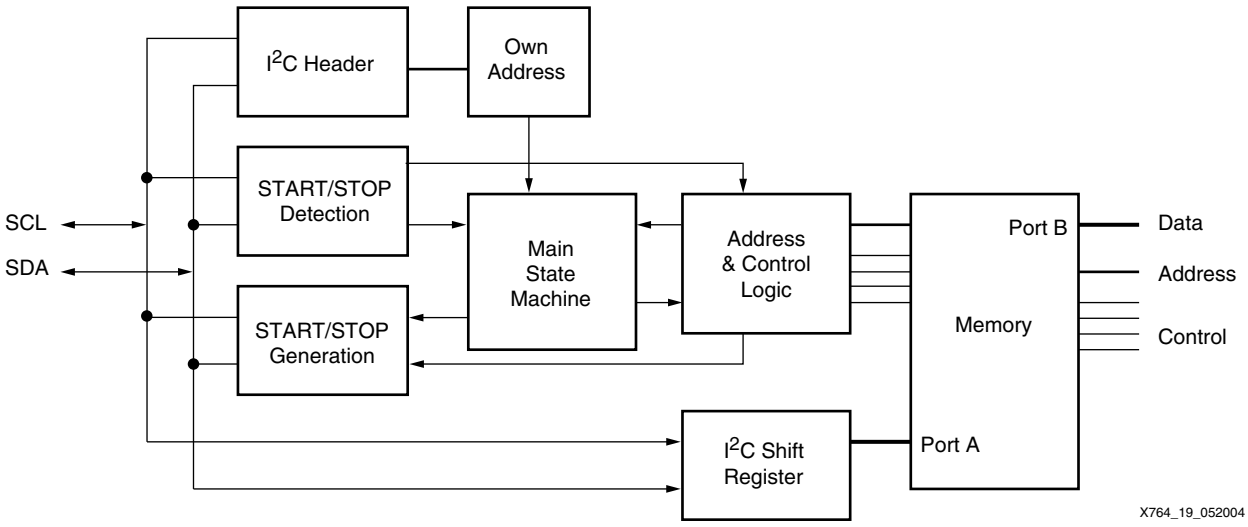

Figure 18: **Sample I²C Design**

The reference design stores the values of the different TZA registers in memory. At startup, these values are shifted to the TZA device through the I²C connection. The only functionality used from the I²C is Master Write mode.

The initial configuration of the transceiver register settings can be done via the supplied PC software from Philips. This software allows setting of transceiver register bits through an I/O port of the PC. This port takes the function of an I²C port and can modify, read, or write the registers of the TZA device. Implementing this function through the FPGA implies a route through the two I²C control/data lines.

PCB Guidelines

This section alerts the PCB designer to several design issues. Remember that PCB design is not just putting traces on a piece of hard material in order to make electronic components do something meaningful.

Component Placement

Try to place the different circuit components as close as possible to each other and line them up with regard to the I/O pinning. Position the components so that PCB traces do not need to take a lot of turns, corners, and pass-through PCB vias when going from one component to another.

The FPGA is very flexible for this because any pin can provide any required functionality. When designing with the TZA optical transceiver, an extra advantage is given through the flexible pin assignment of this component (bus swap, clock polarity switching, DDR enable, etc.).

A straight, short connection improves all possible parameters of a PCB layout:

- Signal integrity
- Transmission line effects
- Capacitance and inductance
- Operating frequency

Many peripheral components can be designed inside an FPGA, as shown in Figure 19, improving the total design performance.

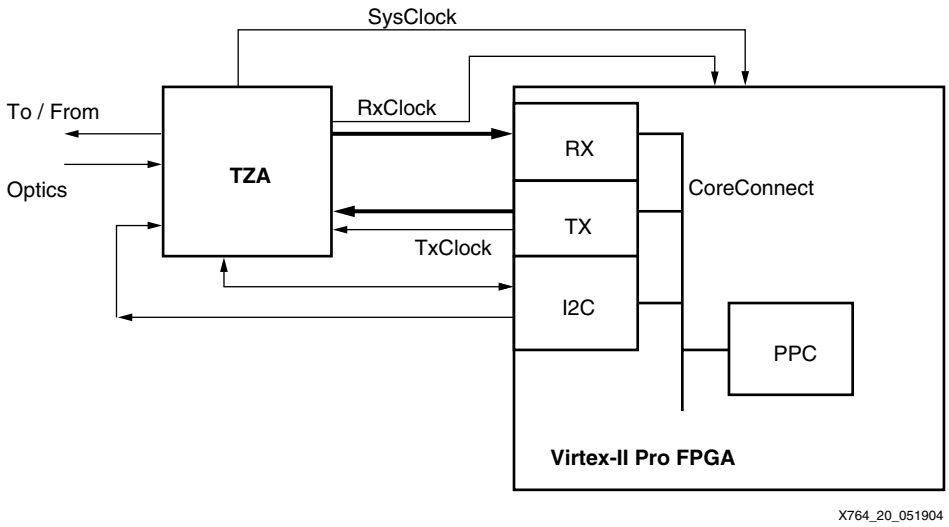

Figure 19: **TZA and Virtex-II Pro FPGA Design Example**

When distances are long and transmission line effects matter, make sure that all transmission lines are terminated properly to control reflections.

Before starting a PCB layout, determine the number of layers to be used. Decide also the destination of the layers (signal, ground, power, and so forth).

It is best to take an extra day with the layout software to shuffle the components on your PCB and find a good placement, rather than risk ending up with a bad PCB design.

Component Placement Examples

Figure 20 and Figure 21 show two possible component interconnection schemes.

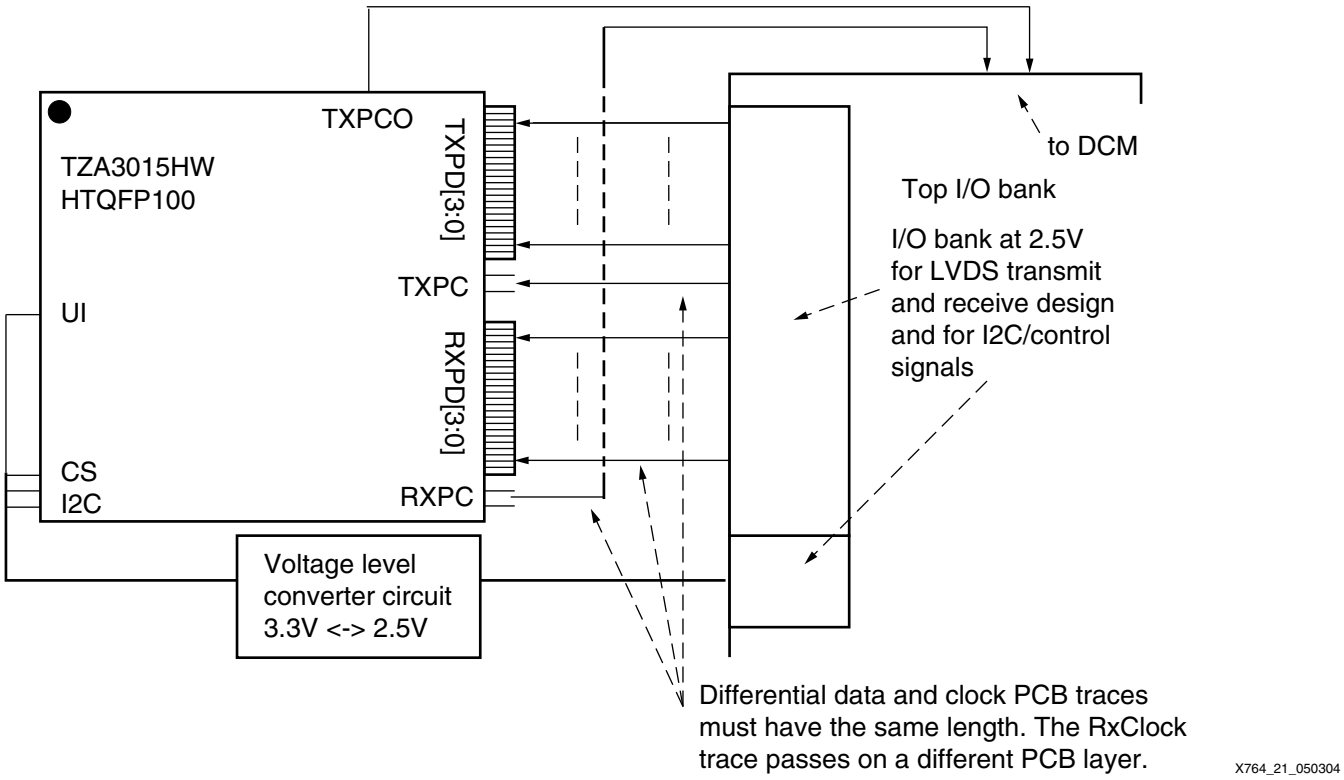

Figure 20: **TZA Connected at Bank 7 of Virtex-II and Virtex-II Pro FPGAs**

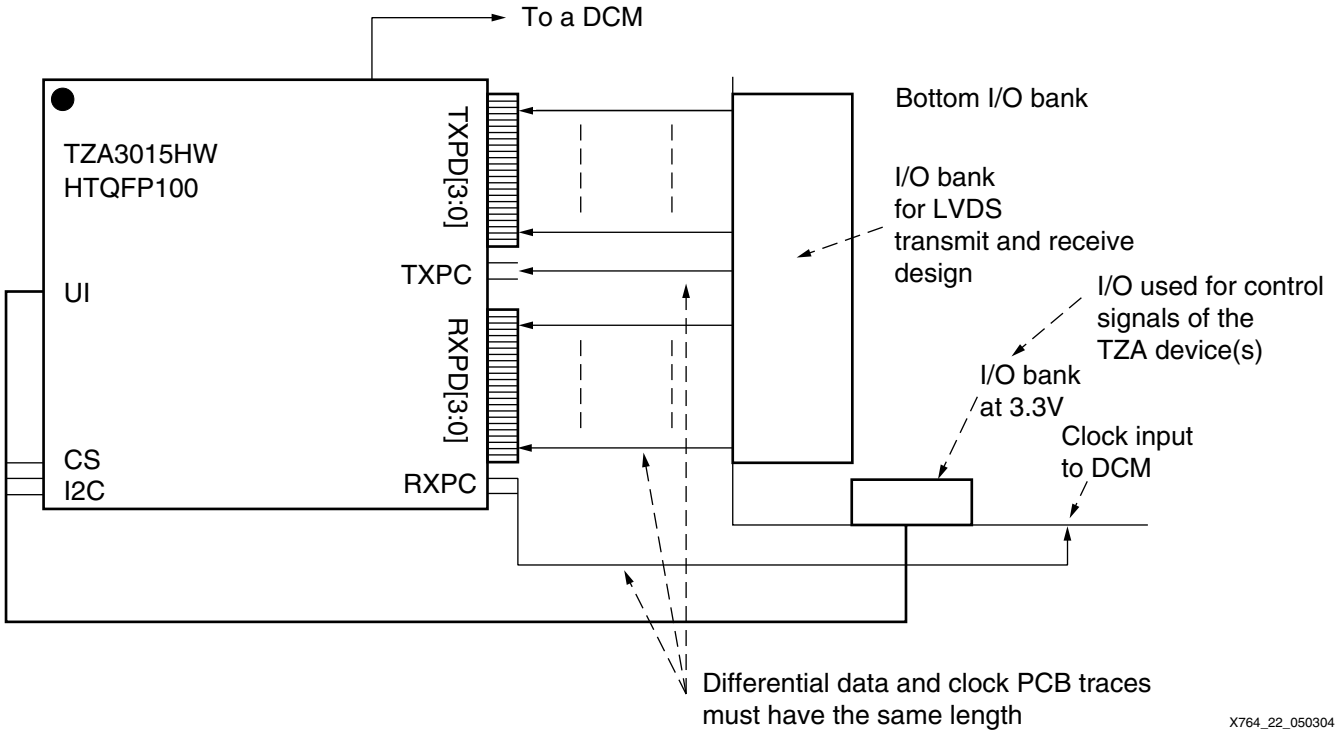

Figure 21: **TZA Connected at Bank 6 of Virtex-II and Virtex-II Pro FPGAs**

Make sure all Low Voltage Differential Signaling (LVDS) transmission lines have the same length! It is recommended that the clock and data signals for both transmission and receive parallel data buses have the same length. The DCM in the Virtex-II or Virtex-II Pro FPGA provides the necessary phase shift.

PCB Guidelines and Recommendations

The following bulleted list summarizes the key guidelines for PCB designs:

- Spend a sufficient amount of time when placing components for layout. With today's layout tools, this is a kind of shuffling block game.
- Keep the trace lengths as short as possible.
- Spend time determining the number of PCB layers and how the layer stack-up will be realized.
- If possible during PCB layout, keep the length of a track shorter than the travel and reflection time of the signal on the trace. If not possible, take the transmission line theory into account.
- Route traces on the PCB as far from each other as possible to avoid or minimize crosstalk effects.
- Do not route traces into 90 degree turns, and never route traces into 180 degree turns (except when you know what you are doing). 90 degree turns increase the effective width of the trace, contributing in parasitic capacitance. At very fast edge rates (˜100 ps), these discontinuities can cause significant signal integrity problems.
- Use round, circular turns. If this is not possible, use 45-degree corners.
- Take the guidelines of the signal return paths into account.
- Guard traces.
- Remember the importance of ground planes.

- Pay special attention to the power distribution. Make sure the ground and power planes are well connected with many bypass capacitors to create low-inductance signal return paths.
- For dead-end stubs, never allow a trace stub to exist without termination. This kind of stub is an antenna, and the uncontrolled impedance causes signal reflections that result in unpredictable behavior.
- For PCB trace turns, never make a 180-degree trace turn on a PCB. This turn makes a perfect antenna. A board should never have traces that turn more than 45 degrees.

Note: Be aware that antennas work in both directions; they emit and receive signals.

- When dealing with transmission lines and using free available calculation programs, be aware that not all published formulas give the same result. Most formulas are empirically put together and then fine-tuned by others or even more experimental work.
- Take the average out of different calculations, done with different programs and tools.
- As with all things, experience helps! If you have no experience at all, try to find somebody with some or take some good training courses.

Reference Design

The reference design uses techniques described in XAPP291, XAPP265, and XAPP268 (see "Reference Material"). The design is set up as a modular block design. Each block has its own functionality as shown in Figure 19.

Design Files

A complete functional design is included in the ZIP file with this application note. The reference design files can be downloaded from:

http://www.xilinx.com/bvdocs/appnotes/xapp764.zip

The files all refer to the Xilinx XLVDSPro demonstration board.

The BIT file for download of the design is for JTAG use only! Extra design files are included so that the design can easily be used in other applications and systems. Extra files relate to:

- I^2C interface
- Register for pre-programmed and control lines
- Large receive FIFO, the default one is 16 places deep
- UCF (User Constraint File) and NCF (Netlist Constraint File) showing the parallel data inputs and outputs place at same side of the FPGA (in the same bank of I/O)

Design Directory Setup

This section provides the directory setup of the reference design files.

/Design

/Ise

/Mentor

/Synplicity

/DdrReceiver

/DdrTransmitter

/DdrToplevel

/SimScripts

Directory for all simulation scripts

/Simulation

Directory where all simulation execution related files are stored

/Synthesis

/Mentor

/Synplicity

/DdrReceiver

/DdrTransmitter

/DdrToplevel

/Vhdl

/AppDesignAndControl

/DcmPhaseCtrl

/DdrReceiver

/DdrTransmitter

/Toplevel

/TzaI2Ccontroller

/LargeRxFifo

/Extra

NCF and UCF files for transmitter and receiver placement in an XC2VP20-FF896

/Ucf

Basic UCF file

/Xps

Not used in this design.

/Documents

Documentation files

Reference Material

The following documents provide supplementary material useful with this application note:

Designs:

- Xilinx XAPP265: "High-Speed Data Serialization and Deserialization (840 Mb/s LVDS)"
- Xilinx XAPP268: "Active Phase Alignment"
- Xilinx XAPP291: "Self-Addressing FIFO"

PCB:

- Xilinx XAPP623: "Power Distribution System (PDS) Design: Using Bypass/Decoupling Capacitors"
- Xilinx XAPP659: "Using 3.3V I/O Guidelines in a Virtex-II Pro Design"
- *RF Circuit Design* (theory and applications): Reinhold Ludwig/Pavel Bretchko (Prentice Hall ISBN 0-13-095323-7).
- RFI/EMI/EMC: *A Designer's Handbook*: Gary A. Breed
- *Microwave Circuit Analysis and Amplifier Design*: Samuel Y. Liao (Prentice Hall ISBN 0-13-586736-3)

- *Kluwer Technische Boeken*: Electronica Vademecum: Kluwer Deventer-Antwerpen, 1980
- *Reference Data for Engineers*: Edward C. Jordan (Howard W. Sams & Co.)
- *Transmission Lines for Digital and Communications Networks*: Richard E. Matick (McGraw Hill, 1969)
- IPC publications: IPC-2141 and IPC-D-317A, along with corrections and adjustments to these publications from IPC and others
- *Radio Handbook*, 23rd edition: William I. Orr (Newnes, 1997. ISBN 0-7506-9947-7)
- *High-Speed Digital Design*: Howard Johnson – Martin Graham (Prentice Hall ISBN 0-13-395724-1)
- *High-Speed Signal Propagation*: Howard Johnson – Martin Graham (Prentice Hall ISBN 0-13-084408-x)

Conclusion

A Xilinx Virtex™-II or Virtex-II Pro™ device can easily connect to a Philips TZA3015HW 30 Mbit/s to 3.2 Gbit/s A-rate 4-bit fibre optic transceiver as described in this application note.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
05/25/04	1.0	Initial Xilinx release.