XAPP626 (v1.1) April 30, 2002

High-Speed Interface with a Velio SerDes

Author: Mike Dauber (Velio) and Marc Defossez (Xilinx)

Summary

This application note describes the design of an interface between a Xilinx Virtex™-II FPGA and a Velio Communications SerDes device. The reference design specifically uses a Velio VC1003 and a Xilinx XC2V1000 FPGA.

Introduction

The Velio VC1003 is a high-performance SerDes device capable of 3.125 Gb/s data rates with a total power dissipation of only 1.9 W. The solution described in this application note uses a Virtex-II XC2V1000, 456-pin BGA package, -4 (speed grade) FPGA. The reference design has four channels connected to the FPGA. The Velio Communications VC1003 data sheet gives a detailed description of the SerDes device and interface signals. The VC1003 data sheet is available on the Velio web site at http://www.velio.com.

Virtex-II Device Overview

The Virtex-II XC2V1000 FPGA sends or receives four channels of parallel un-encoded data (eight bits plus one control bit per channel) to or from the Velio VC1003. On the receive side each channel has a clock; while each transmit channel is clocked from TC03.

The Virtex-II architecture includes three basic sections: the Parallel Port Interface (PPI), the TX logic, and the RX logic. In this application note, the PPI is not described in detail (Velio uses a parallel port to communicate between a PC and the board). The TX logic is essentially some RAM (to generate patterns), connected to the **Serializer**, connected to the Input/Output Blocks (IOBs). All the TX logic is run from the TX clock. The TX clock also drives some of the RX logic. The RX logic is described in more detail in the **Receive Path** section. Figure 1 is a basic Virtex-II block diagram showing the clock distribution with respect to the overall design.

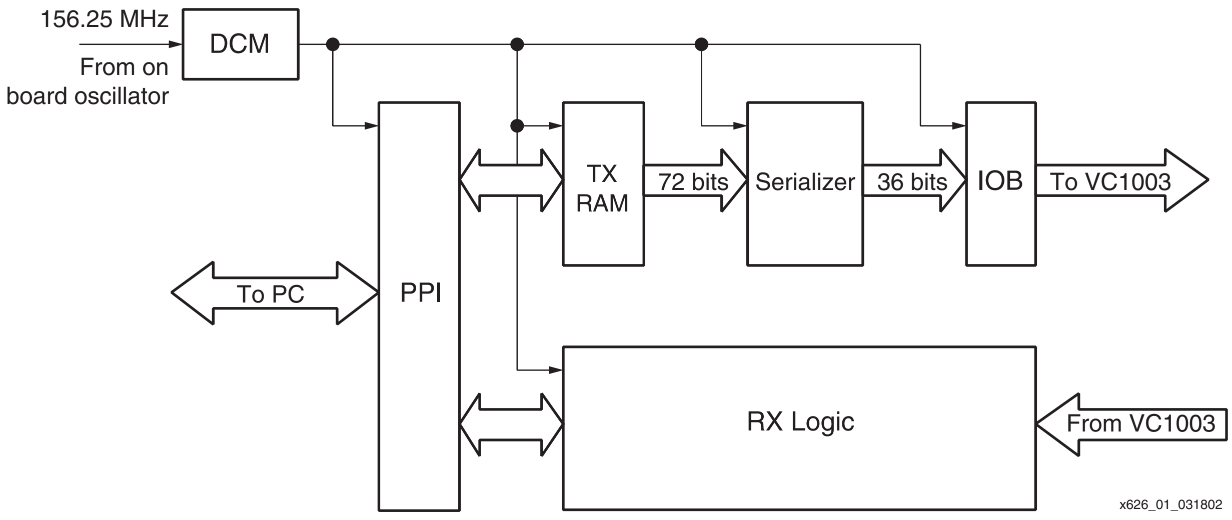

Figure 1: **Virtex-II Block Diagram**

Receive Path

A block diagram of the RX logic is shown in Figure 2. There is a 156.25 MHz DDR interface between the VC1003 and the input to the Virtex-II FPGA. After passing through a Xilinx Digital Clock Manager (DCM), the local receive clock latches the data from the SERDES into the DDR registers. The DCM allows each lane to be individually tuned ensuring correct data is latched by phase-shifting the clock in very small increments. The data is then read into a deserializer (**Deserializers**). The data from the deserializer is then read into a FIFO. The data transfers from the local receive clock into the TX clock domain.

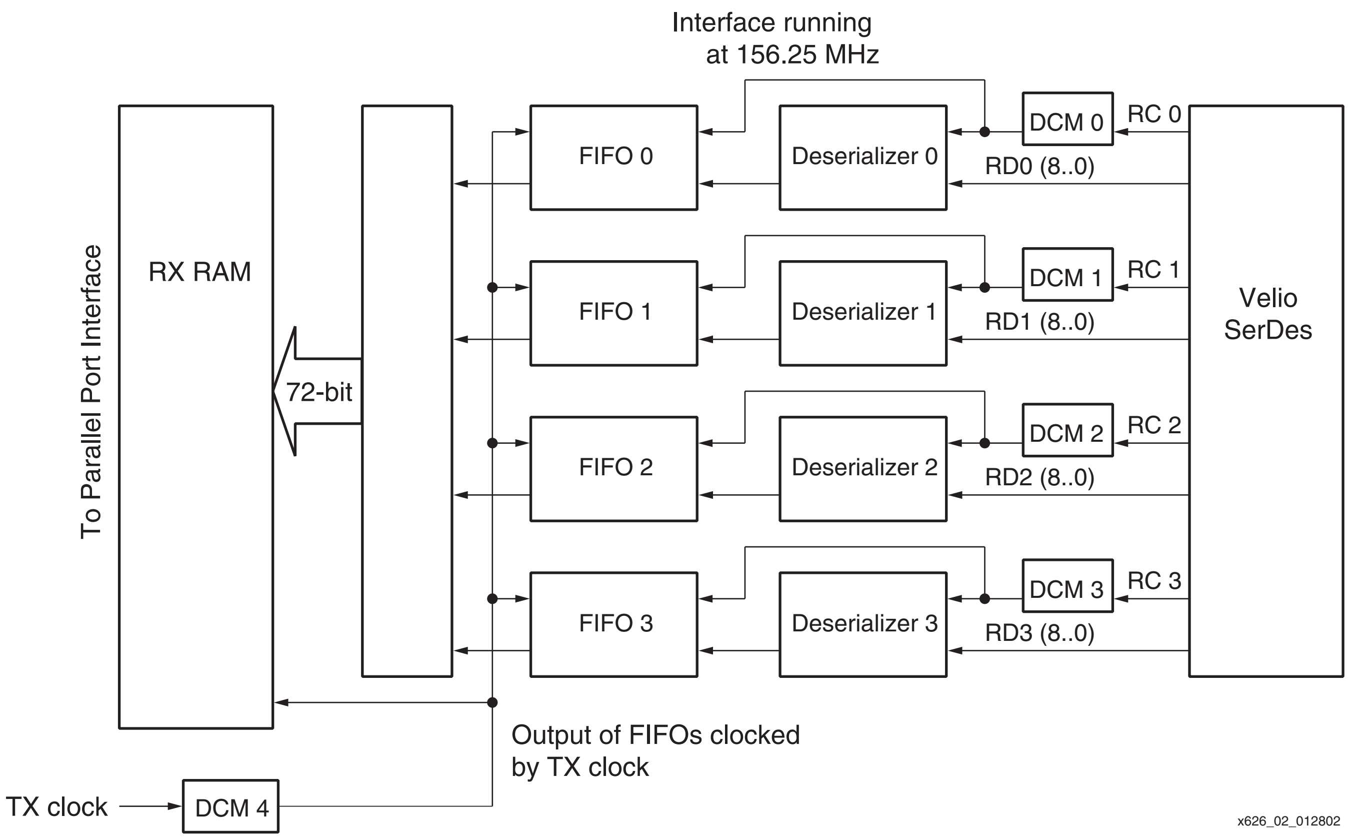

Figure 2: **RX Logic Block Diagram**

The TX clock should be cleaner than the local receive clock since the TX clock is provided by an on board oscillator. The local receive clock contains clock jitter generated by the VC1003 plus the jitter added by the DCM. The DCM jitter is additive. The FIFOs also allow all the data to be clocked off of the same clock domain. Reads and Writes into RAM (or whatever internal logic is being used in a particular design) utilizes the same clock. To minimize clock domain crossing points and make designing easier, the Virtex-II device runs off of the TX clock wherever possible.

After the FIFOs, the data goes into a 72-bit pipeline register (four channels, 16 data bits per channel for both the rising and falling edge data, and the two control bits per channel). The contents of the register are read into RAM. The RAM is controlled by a state machine and is not detailed in this application note.

Deserializers

The deserializer allows the FPGA to latch the correct data from the VC1003. The deserializer has the DDR registers packed into the IOBs of the FPGA (an explanation of packing the IOBs is described in the **Design Constraints** section). The data latched on the falling edge of the clock has only a half cycle to get setup before the data must be latched by the register feeding the FIFO. At 156.25 MHz DDR a half cycle is only 3.2 ns.

It is somewhat difficult for the place-and-route (PAR) tool to accomplish automatic place and route without hurting the timing somewhere else in the design. To compensate for this, the deserializer delays the falling edge (called *Data_N* in Figure 3) data by a cycle-and-a-half and the positive edge data (*Data_P*) by a cycle. This means that the PAR tool only has to meet a 9.6 ns register-to-register constraint on the *Data_N* path instead of 3.2 ns (the *Data_P* is still 6.4 ns). It is necessary to add a constraint in the user constraints file (`.ucf`) to get these PAR savings (see **Design Constraints**).

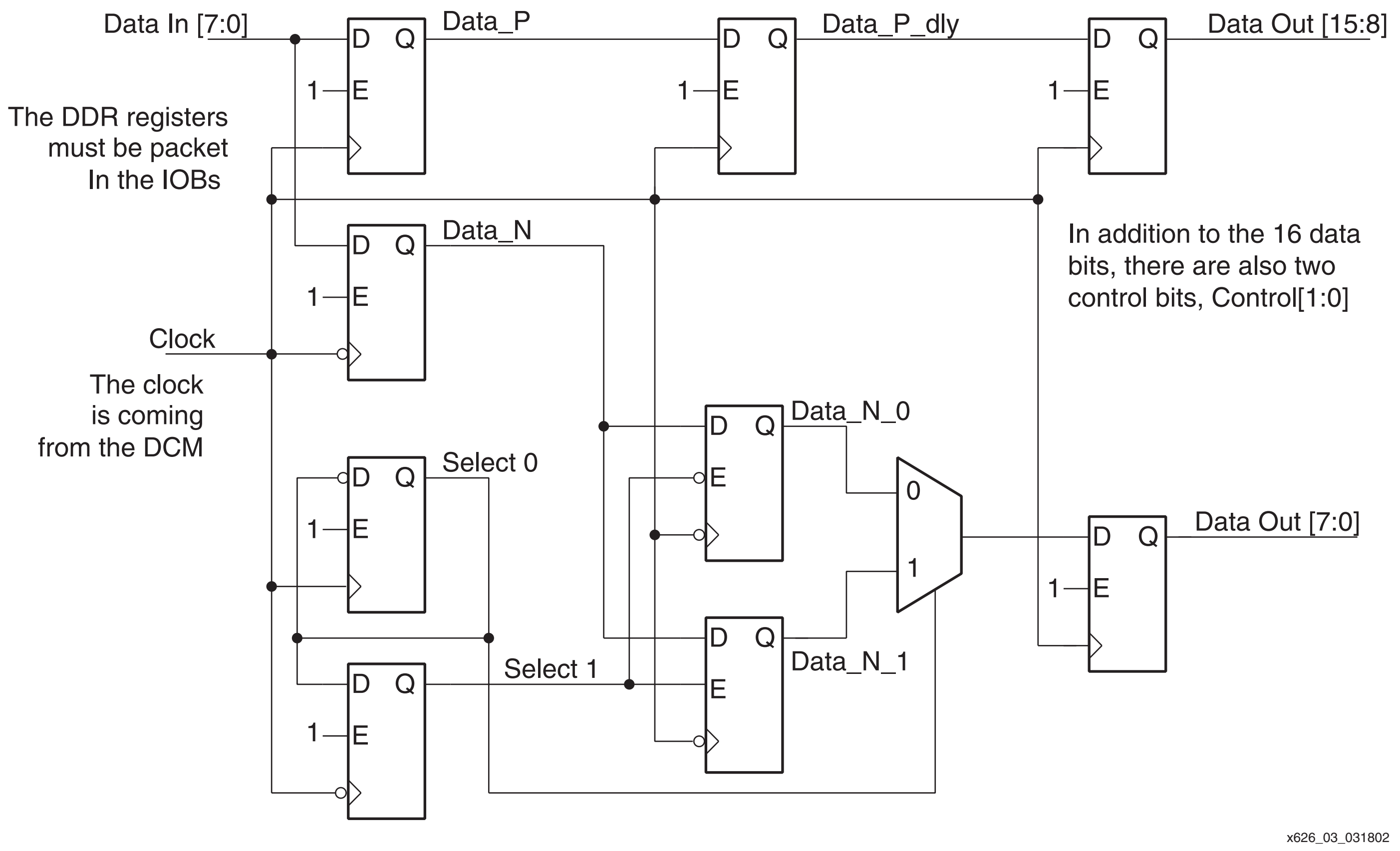
Data In [7:0]
Data_P
Data_P_dly
Data Out [15:8]
D Q
E
1
The DDR registers must be packet In the IOBs
Data_N
In addition to the 16 data bits, there are also two control bits, Control[1:0]
Clock
The clock is coming from the DCM
Data_N_0
Select 0
0
1
Data Out [7:0]
Select 1
Data_N_1
x626_03_031802

Figure 3: **Deserializer Module Block Diagram**

The lower half of the deserializer clocking *Data_N* can be confusing (Figure 3). Essentially, the *Select_1* signal enables only one of the two flip-flops at a time. After it is enabled, a flip-flop waits until the next clock cycle before its output is selected by the multiplexer. The setup and hold is 9.6 ns versus the 3.2 ns that would otherwise be available. A timing waveform is shown in Figure 5. The deserializer is the first waveform.

Serializer

The serializer (Figure 4) is very similar to the deserializer, except it does the exact opposite function. In the deserializer, data is pumped into the DDR registers, the deserializer divides it into data, data is then divided into data clocked off of the rising edge (*Data_P*), and data clocked off of the falling (*Data_N*). In contrast to the deserializer there are two kinds of data in the serializer. The top nine bits are *Data_H*, data clocked off of the falling edge. The bottom nine bits are *Data_L*, data clocked off of the rising edge. The serializer is needed to handle the 3.2 ns register-to-register delay accounted for by the PAR tool. Between the output of a register right before the DDR register, clocked by rising edge of the clock, and the input to the DDR register, latching on the falling edge of the clock (when not using the serializer). The serializer allows the 9.6 ns constraint to be placed on the *Data_H* path. This constraint is entered in the same manner as the constraint is entered for the deserializer, by using Slow/Fast exceptions (**Design Constraints**). The clock sent along with the data is just a DDR register with a "1" and a "0" tied to its inputs. In the reference design the DDR registers are instantiated using the FDDRSE primitive. Figure 5 shows two timing waveforms. The serializer is the waveform on the bottom.

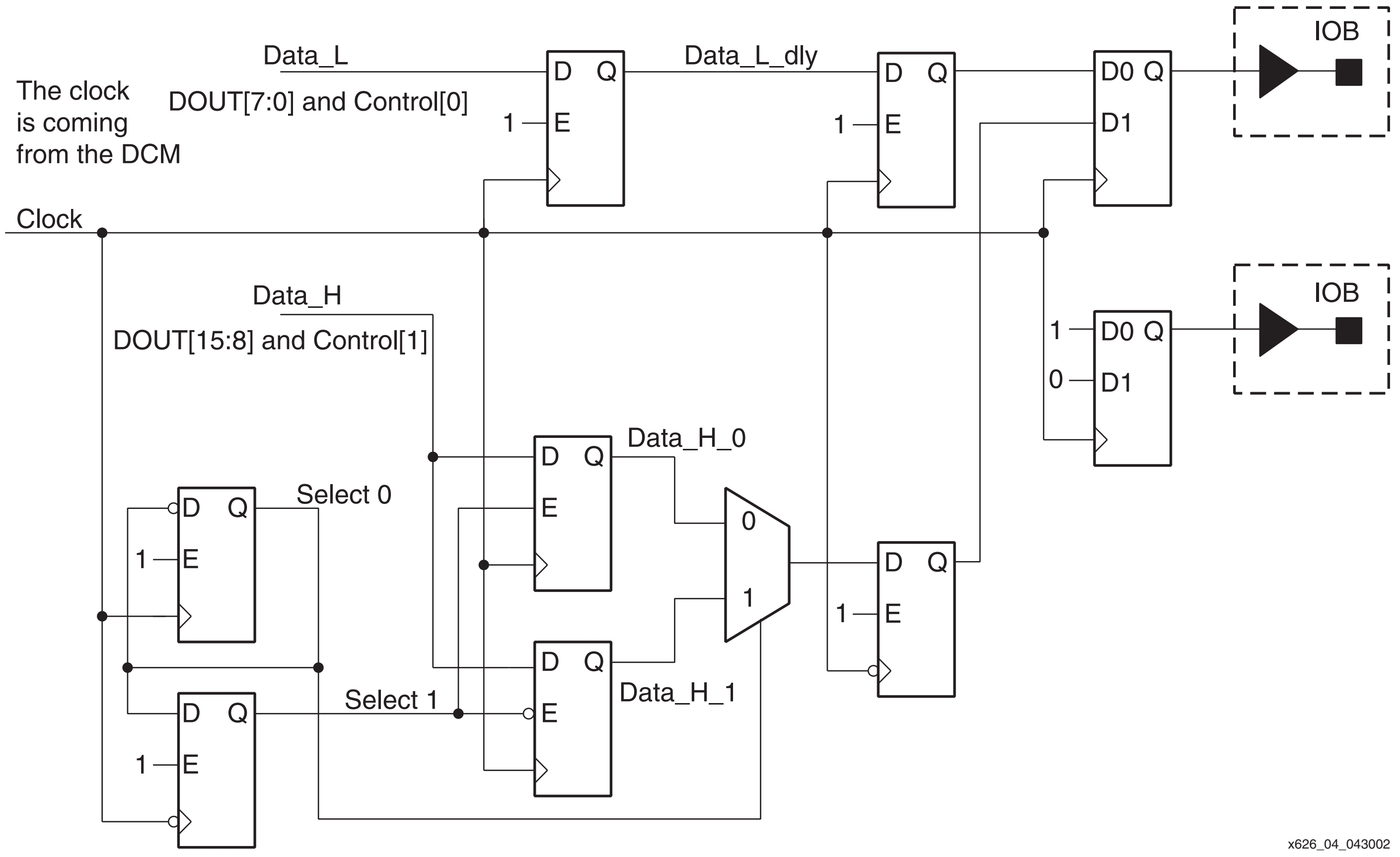

Figure 4: **Serializer Module Block Diagram**

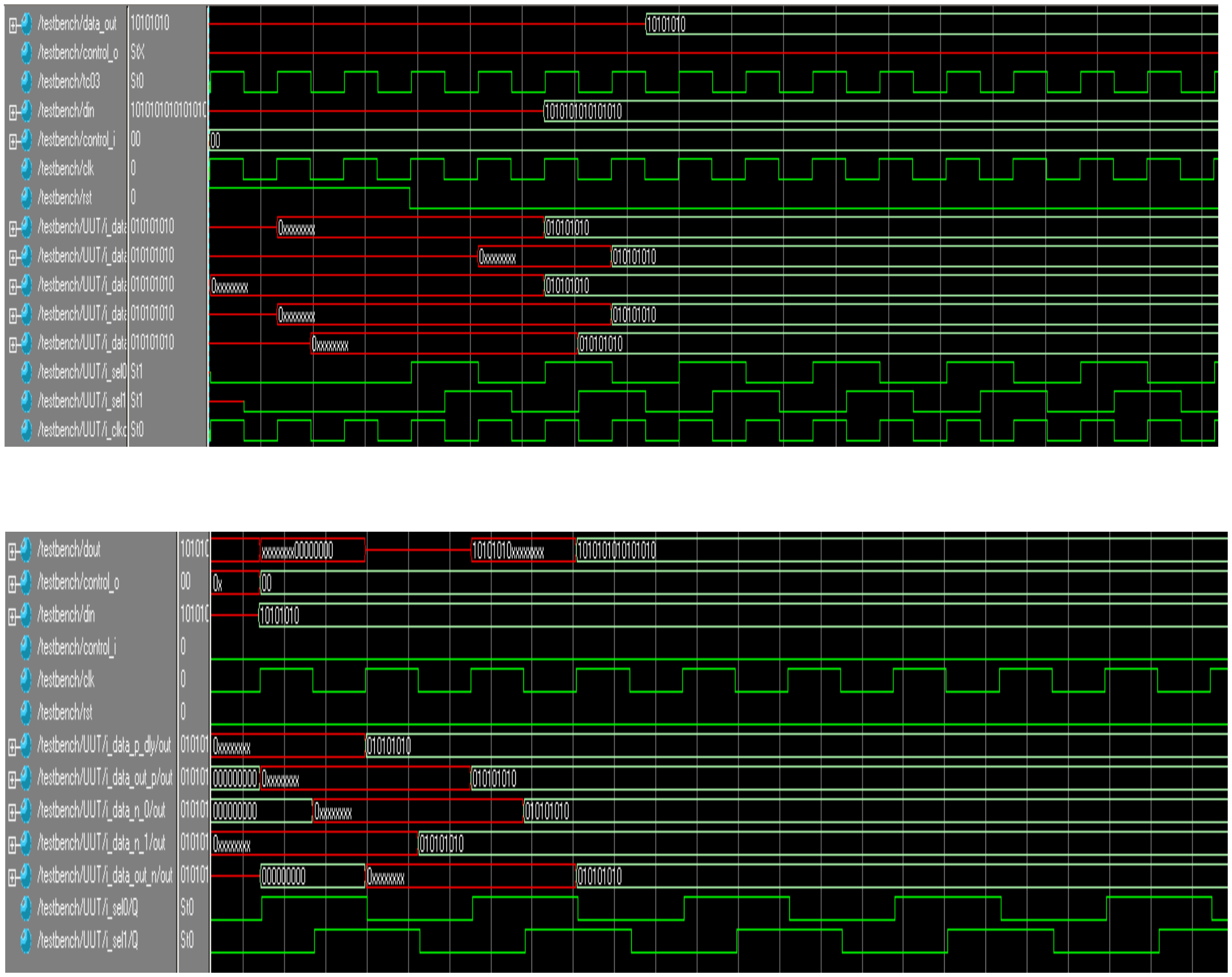
/testbench/data_out 10101010
/testbench/control_o StX
/testbench/tc03 St0
/testbench/din 1010101010101010
/testbench/control_i 00
/testbench/clk 0
/testbench/rst 0
/testbench/UUT/i_data 010101010
/testbench/UUT/i_sel0 St1
/testbench/UUT/i_sel1 St1
/testbench/UUT/i_clk St0
/testbench/dout
/testbench/control_o
/testbench/din
/testbench/control_i
/testbench/clk
/testbench/rst
/testbench/UUT/i_data_p_dly/out
/testbench/UUT/i_data_out_p/out
/testbench/UUT/i_data_n_0/out
/testbench/UUT/i_data_n_1/out
/testbench/UUT/i_data_out_n/out
/testbench/UUT/i_sel0/Q
/testbench/UUT/i_sel1/Q
x626_05_012802

Figure 5: **Simulation Results of the Serializer/Deserializer Circuits**

Design Constraints

The reference design requires a number of constraints (in addition to pin constraints). Many are specific to the reference design and are important to note. The easiest way to enter constraints is by launching the Xilinx Constraints Editor from either Project Navigator or Design Manager after having run the "Translate" step in the design flow.

1. Global Clock Constraints
   a. On the "Global" tab, all the clocks associated with data going to or from the VC1003 should have a clock period set to 6.4ns and a duty cycle of 50%.
2. Packing the DDR Registers in the IOBs
   a. On the "Misc" tab click on the "Specify" button under "IOB Register Control".
   b. In the field next to "filter" enter part of the name of the output of the DDR register (so, in this case *Data_P* or *Data_N*).
   c. Select all of the signals (in the case of this design there were 72, one for each bit out of a DDR register) that should be in DDR registers, click "Add" and then "OK".
   d. The DDR registers should automatically be packed in the IOBs (only the inputs from the VC1003 should be packed in the IOBs. The output to the VC1003 should not be packed in the IOBs.).

   e. To check that this occurred, go into FPGA Editor and look at the cell where the input pin is located (this is also the location of the IOB). Expand the cell by double clicking on it and check to ensure that the DDR input register (lower right-hand corner) is being used.

3. Relaxing the Clock Constraint on *Data_N* to 9.6 ns:

   a. On the "Advanced" tab click on "Create" next to "Group Elements By Instance Name".

   b. Use the filter to find the name that the synthesis tool gave to the signal (searching on *Data_N* should work), and add all of the signals to the list.

   c. Give them a name (*slowdown_data_n*) and specify the design element as "FFs" and click on "OK".

   d. Click on the "Specify" button next to "Slow/Fast Exceptions (FROM TO)".

   e. In the field called "TIMESPEC Name" enter `TS_slowdown` (it can be anything that starts with "TS").

   f. In the "FROM" group pull-down menu select whatever was entered in the "Group" command (*slowdown_data_n*), and in the TO select "All Flip Flops".

   g. Under "Type" click on the field that says "Explicit" and enter 9.6 ns into field, then click "OK". This tells the PAR tool not to put the 6.4 ns constraint on this path (since it only has to be 9.6 ns, making it 6.4 ns wastes effort on the tools part, and increases the chances that timing will fail somewhere else), and extend it to 9.6 ns.

4. General rules

If there are places where 6.4 ns is not needed, either specify a new, slower FROM TO to leave slack where the timing is more critical, or use one of the two "FALSE PATHS" to specify someplace where the tool can ignore timing all together (this varies from design to design).

A flip-flop clocked by TX clock going to control pins in the VC1003 does not have a 6.4 ns constraint because these signals do not change as fast as the real data signals.

DCMs

Digital Clock Managers (DCMs) are an important feature of the Xilinx Virtex-II architecture. However, special care must be taken when using the DCM. The jitter on any clock coming into the DCM will have jitter added to it at the output. In addition, different numbers of DCMs are provided depending on the component used. For the XC2V1000, eight DCMs are provided. For some device versions that number can reach 12, for others, that number is only four.

Conclusion

The VHDL and Verilog code for the serializer and deserializer designs are available in the downloadable file **xapp626.zip**. The reference design also calls smaller sub-modules.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/18/02	1.0	Initial Xilinx release
04/30/02	1.1	Removed sentence "DDR FFs must be made in CLB" from Figure 4.