XAPP265 (1.1) November 7, 2001

# High-Speed Data Serialization and Deserialization (840 Mb/s LVDS)

Author: Nick Sawyer

## Summary

The serial transfer of data between cards on a backplane is often a requirement in digital system design. Serializing the data makes greater use of the available resources (pins). A system processing 64 bits of data at 80 MHz can use a 64-bit data link (with control signals) to another device or across a backplane. This same data can be transmitted over 16 lines SDR at 320 MHz (producing a 75% pin savings). Alternatively, the data can be serialized to eight differential pairs (still 16 lines) running at 320 MHz DDR (640 Mb/s). This also produces a pin savings of 75% with greater signal integrity and lower power.

This application note addresses circuits capable of transferring up to 16 data channels at up to 840 Mb/s each for an aggregate data transfer per link of over 13 Gb/s. The design may be used multiple times in a Virtex™-II device. There is no limit to the number of transmitters (within pinning constraints) that can be used if they are using the same transmission clock. The Digital Clock Managers (DCMs) available to the designer, a maximum of 12 depending on device size, limit the number of receivers.

## Introduction

Signal integrity at high speeds such as 840 Mb/s is a big issue, prompting designs using differential signaling. Although doubling the number of pins used, this still results in 80% pin savings. In addition, designing a system to work synchronously at high speeds is virtually impossible. Using a clock forwarding technique, the data, clock, and possibly a framing signal are forwarded together along the backplane. The clock frequency used is normally half the data transmission frequency, which means that data changes on each edge of the clock, providing so-called Double Data Rate (DDR) signaling. The advantage of this method is that the harmonic content of the clock and data lines is the same. Another solution for data transfer is to embed clock information in the data stream at the transmitter, and then recover the clock and data at the receiver. This is covered in a separate application note. All the files for both design and implementation discussed below are available from the Xilinx web site. When synthesizing the files, the synthesizer must retain hierarchy. And it should be remembered that the I/O and global buffers have already been instantiated. It is usually best to turn off further global buffer insertion in the tool.

## Board Level

One possible system architecture is shown in Figure 1. The master card interfaces to *m* slave cards over a number of *n*-bit wide busses (transmit and receive). Typically, all of these links are of identical width, although this is not a requirement for functionality. The TX and RX boxes shown are from the receiver and transmitter modules.

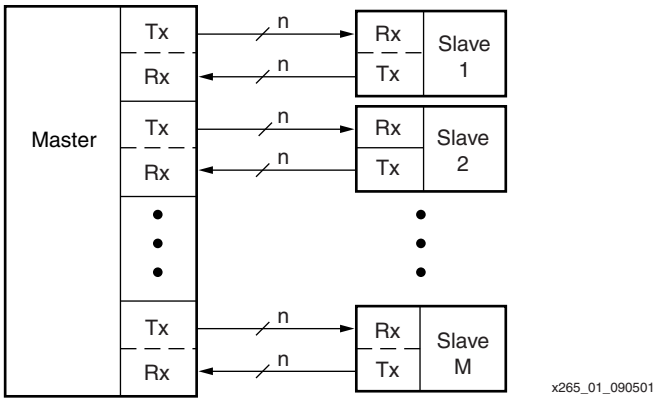


*Figure 1:* **System Configuration**

## Transmitter Module

The transmitter design is relatively straightforward, using several of the unique features of the Virtex-II family. A basic system is defined and shown in Figure 2. The system block processes eight bits of data at a frequency of, for example, 100 MHz (800 Mb/s). The serializer transmits the incoming 100 MHz, 8-bit data words serially on a differential pair using DDR at a frequency of four times the system clock, or 400 MHz (800 Mb/s). A clock is also regenerated for forwarding as is an optional framing signal.

The basic circuit can be “tiled” for larger systems. The files associated with this application note describe a 4-bit building block capable of serializing 32 data lines at 100 MHz to four lines plus clock and frame at 800 Mb/s DDR. Another typical application is to serialize 128-bit internal data to 16 lines plus one clock and one frame signal for 10+ Gb/s signaling. This is easily achieved using four 4-bit blocks, each of which is effectively standalone by design. When tiling, only data lines are usually added; just one clock and one frame signal (if required) are needed for each data link.

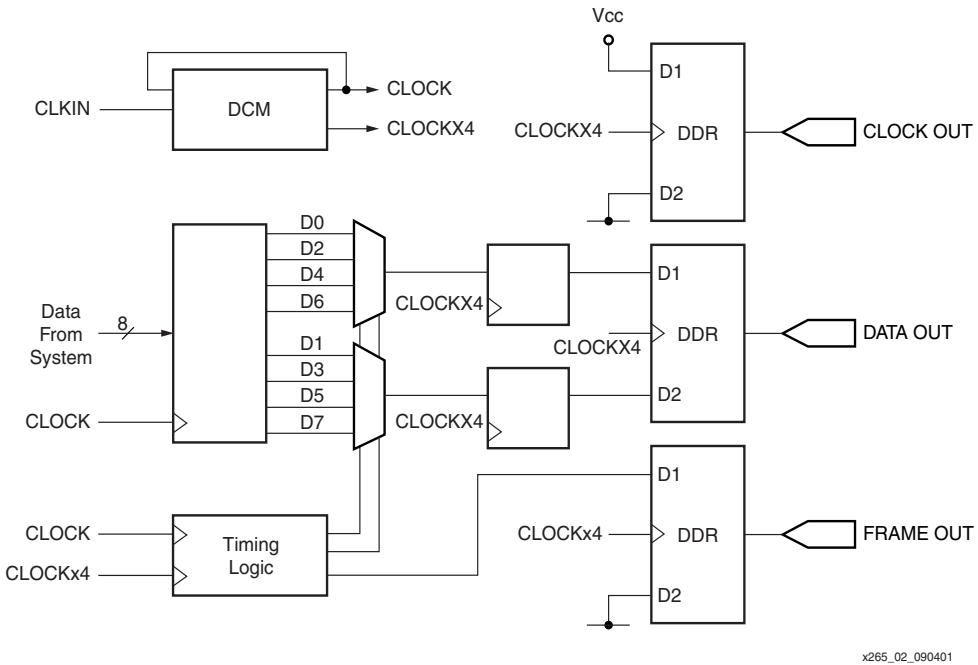


*Figure 2:* **One-Bit Transmitter Block Diagram**

The local clock in Figure 1 is fed to a Digital Clock Manager (DCM) to generate two clocks, where one is exactly four times the frequency of the other as described below. At this point it is prudent to consider any jitter generated by the DCM. Jitter will affect the timing margin of the interface. Three modes of use are possible.

1. Input clock is multiplied by one and four. This solution leads to fairly substantial amounts of jitter (see the Virtex-II data sheet), and is therefore limited to fairly low frequencies of operation (< 400 Mb/s).
2. Input clock is multiplied by two and divided by two. There is less jitter. This solution has been shown to work at frequencies up to 680 Mb/s.
3. Input clock is multiplied by one and divided by four. There is much less jitter. This solution is suitable for use at frequencies up to 840 Mb/s

It is also prudent to look into how the high-speed clocks (CLKX4 and RXCLK) are distributed on the device.

1. Use a single global buffer, and rely on the local inversion available where negative edges are required. Obviously, this method uses less resources. It will also introduce some distortion into the output signals, as the high-low and low-high transitions of the global buffer have some imbalance.
2. Use two global buffers to distribute the high-speed clock and its complement (both are available from the DCM). This method uses more resources. All synchronous elements will be clocked only by rising edges from one of the two distributed clocks. The skew between the two global buffers is very tightly controlled in the silicon giving better results at very high bit rates.

Eight bits of data are registered by the clock signal (CLK). These eight bits are then multiplexed and retimed by the CLKX4 signal (four times the frequency), and then fed to the DDR registers located in the IOBs of the Virtex-II device. The Virtex-II data sheet contains a full description of this functionality. The DDR output is then available at a pair of pins in LVDS signal levels. The $V_{CCO}$ power supply for the I/O bank in the Virtex-II device can be either 2.5V or 3.3V for LVDS functionality.

A transmission clock is regenerated through another DDR register whose inputs are connected to logic High and logic Low. The frame signal is a third DDR register.To ensure the frame signal is a DC balanced line, the output is a logic High four times every eight half-clock periods. The timing diagram for this example is shown in Figure 3. The transmission clock is a copy of CLOCK4, and because it is regenerated inside an IOB, as are the data and frame signals, there is very little skew between these three signals.

The frame signal does not always have to be used because some systems have in-band synchronization, such as 8b/10b coding, built into the data stream.

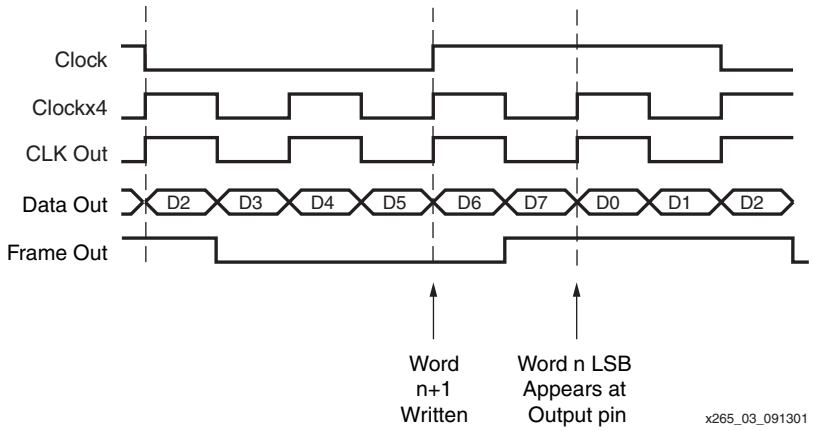


*Figure 3:* **Timing Diagram**

## Four-Bit Transmission (2.5 Gb/s)

The 4-bit transmitter module (serdes_4b_8to1) is written in HDL and is, therefore, fully synthesizable. It contains relative placement information to allow the Virtex-II device to operate at these very high clock speeds. The macro accepts 32-bit data synchronous to a CLK signal and goes on to use this clock and another clock signal (CLKX4) to serialize the data. The two clock signals have to be completely synchronous to one another, so the best solution is to use the DCM capabilities of the Virtex-II device.

For example, an incoming 400 MHz signal could be used to generate 400 MHz for use as CLKX4 and 100 MHz for use as CLK. The output timing characteristics of the 4-bit block are shown in Figure 3. The block has been designed to be used on the right-hand edge of the Virtex-II device, but could also be placed on the left if necessary. The placement of the module is defined by the rloc_origin statement applied to the module in the constraints (**design.ucf**) file used when processing the design through Xilinx implementation tools. This location will obviously vary with device size, but a typical example for the XC2V1000 device would be `"set "tx0/set" rloc_origin = "X58Y0";"` that is, at the bottom (Y = 0) right (X = 58) of the silicon.

The macro is four CLBs high by three CLBs wide in order to have the same pitch vertically as the block RAMS. This is useful if the designer wants to insert a FIFO into the transmission data path. Careful consideration also has to be given to the pins chosen for the transmitter output. Ideally, they will be immediately adjacent to the macro, but a certain amount of spread is acceptable, plus or minus two CLBs vertically will still work at up to 840 Mb/s in a -5 device. The macro physical implementation is shown in Figure 5. This is a useful reference for choosing the pins to be used.

The bit ordering for both the 4-bit transmitter and receiver macros is shown in Figure 4. If a different sequence is needed it is easily accomplished at the top level. For example, if the requirement is to transmit data MSB first, rather than LSB first, the following sequence should be used.

Connect txdata(31) to txmacro datain(0)

Connect txdata(30) to txmacro datain(1)

...

Connect txdata(0) to txmacro datain(31)

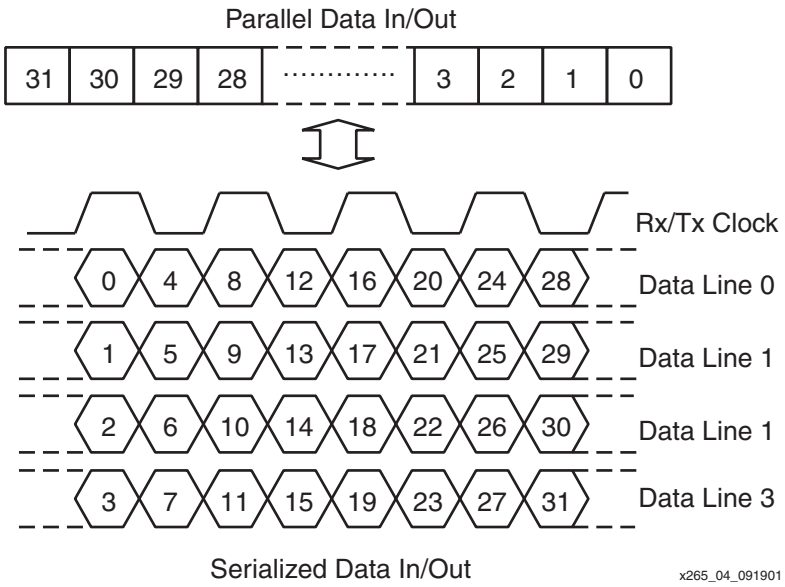


*Figure 4:* **4-bit data formatting**

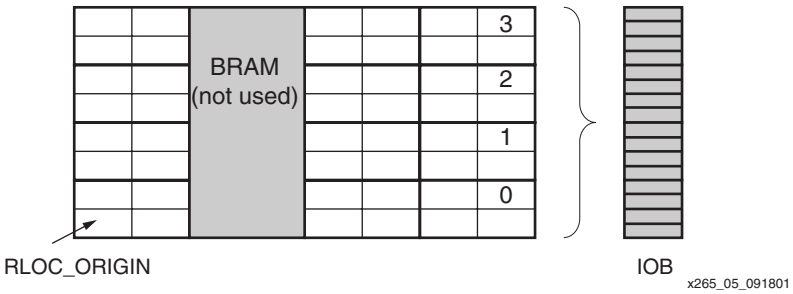


*Figure 5:* **4-Bit Transmitter Macro Physical Implementation**

## Sixteen-Bit Transmission (10 Gb/s)

The 16-bit transmitter module (serdes_16b_8to1) is written in HDL and is, therefore, fully synthesizable. It is made up of four of the 4-bit macros described above and, therefore, contains relative placement information to allow the Virtex-II device to operate at these very high clock speeds. The macro accepts 128-bit data synchronous to a CLK signal and goes on to use this clock and another clock signal (CLKX4) to serialize the data to sixteen output lines.

The block has been designed to be used on the right hand edge of the Virtex-II device, but could also be placed on the left if necessary. The placement of the module is defined by multiple rloc_origin statements. These are applied to the macro via the constraints (design.ucf) file used when processing the design through Xilinx implementation tools. This location will obviously vary with device size, but a typical example for the XC2V1000 device would be:

```
'set "tx0/tx0/hset" rloc_origin = "X58Y0" ;'
'set "tx0/tx1/hset" rloc_origin = "X58Y8" ;'
'set "tx0/tx2/hset" rloc_origin = "X58Y16" ;'
'set "tx0/tx3/hset" rloc_origin = "X58Y24" ;'
```

That is, at the bottom (Y = 0) right (X = 58) of the silicon. Careful consideration again has to be given to the pins chosen for the transmitter output. Ideally, they will be immediately adjacent to the macros, but a certain amount of spread is acceptable. Plus or minus two CLBs vertically will still work at up to 840 Mb/s in a -5 device. A useful reference for choosing the pins to be used is shown in the Figure 7 macro physical implementation. The four individual 4-bit macros do not have to be immediately next to each other, but it is good to place them like this, as the pin-to-pin skew will be minimized. It is also good to ensure that all sixteen data outputs, the clock, and the (optional) frame signal are placed in the same I/O bank. The clock and frame outputs from all four modules are brought up to the top level, and it is up to the designer which of these signals are used. Obviously, only one of the output clocks and one of the frame signals needs to be used.

The bit ordering for both the 16-bit transmitter and receiver macros is shown in figure BB. For example, if the requirement is to transmit data MSB first, rather than LSB first, the following sequence should be used.

Connect txdata(127) to txmacro datain(0)

Connect txdata(126) to txmacro datain(1)

...

Connect txdata(0) to txmacro datain(127)

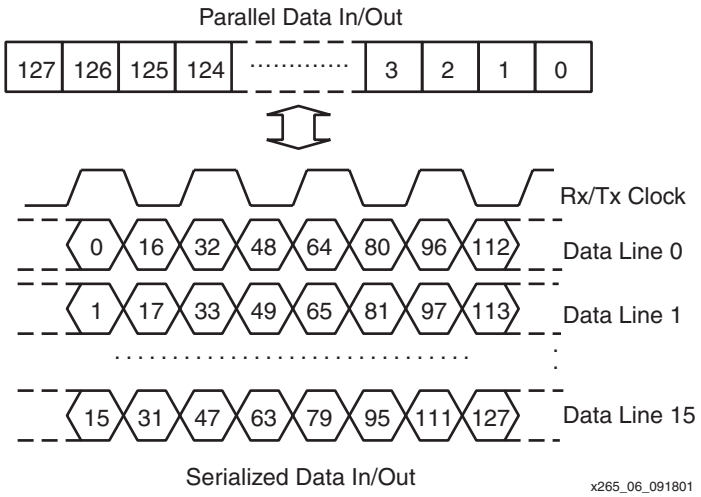


*Figure 6:* **16-bit Data Formatting**

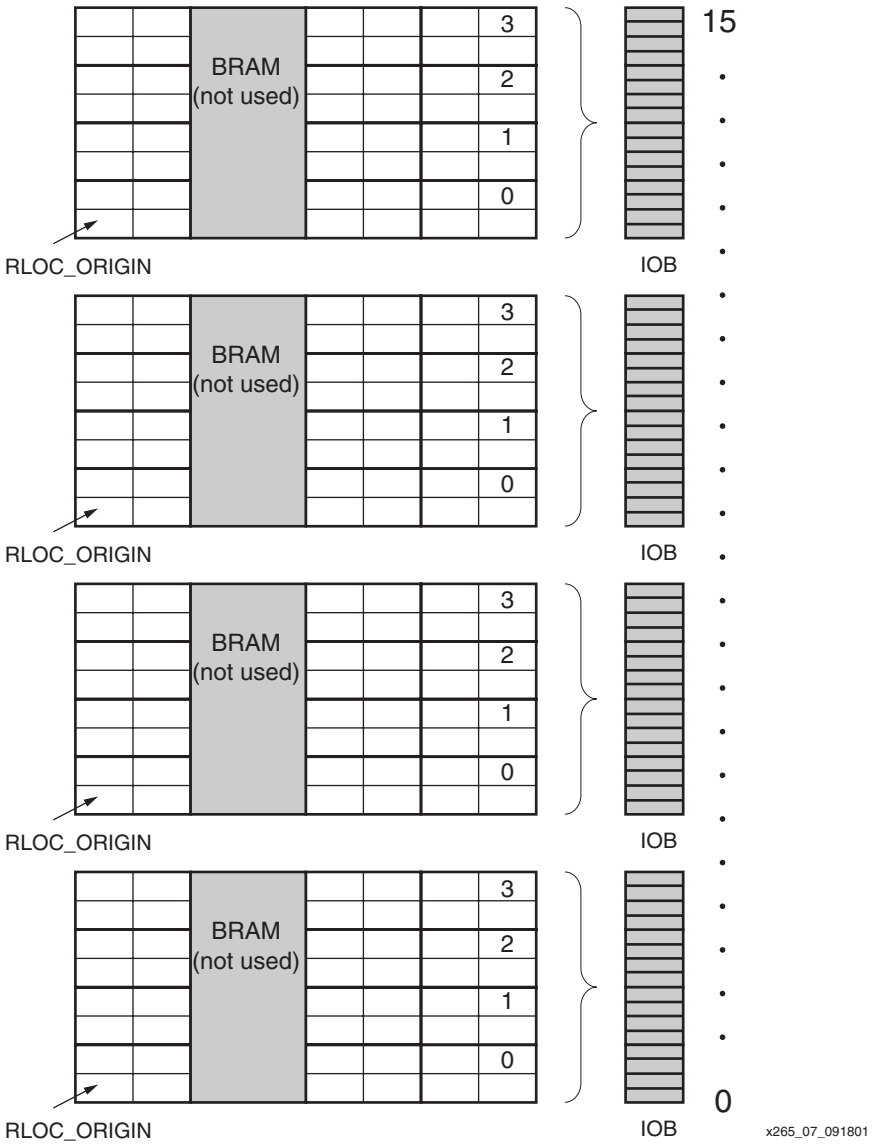


*Figure 7:* **16-bit Transmitter Macro Physical Implementation**

## PCB

The generated differential data, clock, and frame signals need to be routed very carefully on the PCB and the trace lengths strictly controlled. Ideally, all the signals should have the same delay. If the clock trace has a different delay, the receiver DCM can correct it; as a minimum, however, all the data and the frame signals should be matched to within a few tens of picoseconds for the best circuit operation. The physical characteristics of the traces and the PCB are discussed more fully in **XAPP233**.

## Receiver Module

The receiver block diagram is shown in Figure 8. Assuming a single DDR data signal arriving with a clock and a frame signal at 800 Mb/s DDR, it will be deserialized into eight bits of data available via a FIFO at 100 MHz+. The receiver can also be tiled for wider data widths, with only one clock and (if required) one frame signal necessary per data link.

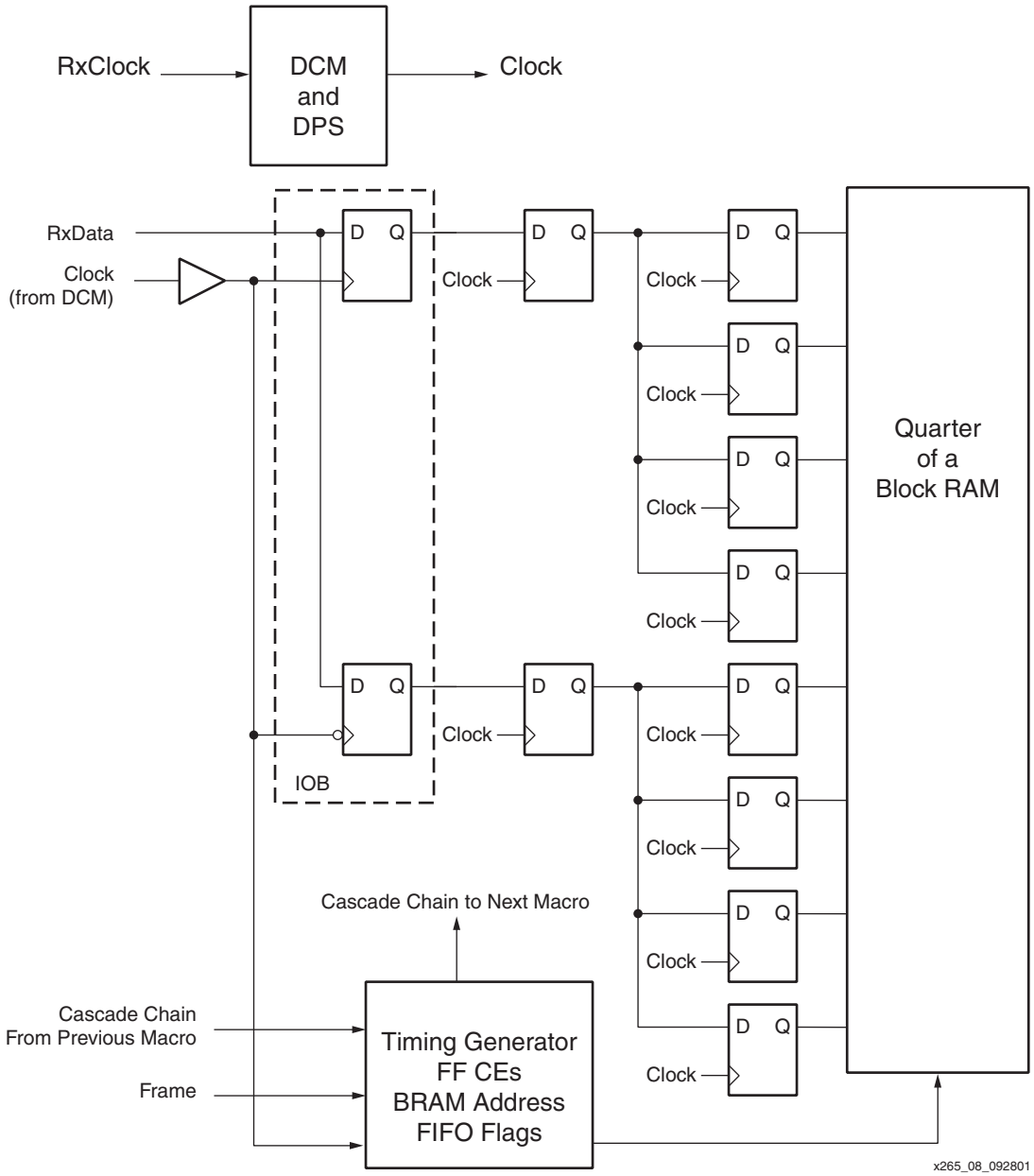


*Figure 8:* **Receiver Block Diagram**

The received clock is fed into one of the Virtex-II DCMs to use the Digital Phase Shift (DPS) feature. The transmitter, normally produces aligned clock and data, and the receiver needs to sample the data in the middle of the data "eye." This is normally achieved by making the clock trace in the link longer by an equivalent of approximately a quarter clock period. Using the DPS allows a "dial-in" for an appropriate amount of phase shift at device power-up. Additionally, the circuit can also be used to remove any differences in clock and data delay that have occurred due to PCB issues. However, *all* the data lines must still have the same delay (track length).

The corrected clock (derived from DCM/DPS through a global buffer) is used to clock the data into the dedicated DDR registers in the Virtex-II IOBs. Two bits of data are captured per clock, one from the negative edge of the clock and one from the positive. These two signals are then reregistered (positive and negative edges) in CLB logic. The same clock is then used to de-multiplex these two lines into eight lines of data. At this point, it is common to build a FIFO into the system, as further processing will probably not occur at a clock rate synchronous to, or even at the same frequency as, the received data. Therefore, the eight received bits are clocked into a very simple FIFO using one quarter of a Virtex-II block memory. These can be configured to be up to 32-bits wide.

The control signals available to the system are very simple, indicating that the FIFO is one-quarter, one-half, or three-quarters full. The frame signal, if used, ensures the serially received data is presented eight bits wide in its original format, and no further framing function is required. Data is clocked into the FIFO every two clock cycles using the received clock and a WE signal, thus avoiding the unnecessarily use of another global buffer.

As mentioned, each received data line in this configuration requires only one-quarter of a global buffer. Four received lines will require one block memory, and sixteen received lines will require four block memories, but still only one DCM/DPS and one global buffer. To make the parallel data wider and slower, or narrower and faster, use serialization factors other than eight to suit the system architecture.

## Four-Bit Reception (2.5 Gb/s)

The 4-bit receiver module (serdes_4b_1to8) is written in HDL, and is, therefore, fully synthesizable. It contains relative placement information to allow the Virtex-II device to operate at these very high clock speeds. The macro accepts 4-bit DDR input data and, optionally, a synchronization signal, synchronous to a RXCLK signal, and it goes on to use this same clock to deserialize the data from a 4-bit DDR to 32-bit wide. The 32-bit data is then written into a FIFO using a block memory. The output of this FIFO together with the necessary control signals are available to the designer for passing data further into the system.

The input timing characteristics of the 4-bit block are shown in Figure 3. The block has been designed to be used on the left-hand edge of the Virtex-II device, but could also be placed on the right if necessary. The placement of the module is defined by the rloc_origin statement applied to the module in the constraints (design.ucf) file used when processing the design through Xilinx implementation tools. This location varies with device size, but a typical example for the XC2V1000 device would be `"set "rx0/hset" rloc_origin = "X0Y0" ;"` that is, at the bottom (Y = 0) left (X = 0) of the silicon.

The macro is four CLBs high by three CLBs wide to have the same pitch vertically as the block RAMS. Careful consideration also has to be given to the pins chosen for the receiver output. Ideally, they will be immediately adjacent to the macro, but a certain amount of spread is acceptable, plus or minus two CLBs vertically still works at up to 840 Mb/s in a -5 device. The macro physical implementation is shown in Figure 9, which is a useful reference for choosing the pins to be used.

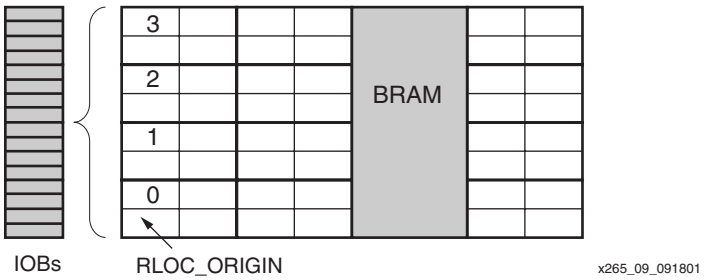


*Figure 9:* **4-Bit Receiver Macro Physical Implementation**

## Sixteen-Bit Reception (10 Gb/s)

The 16-bit receiver module (serdes_16b_1to8) is written in HDL and is, therefore, fully synthesizable. It contains relative placement information to allow the Virtex-II device to operate at these very high clock speeds. The module uses the serdes_4b_1to8 macro mentioned above four times, and accepts 16-bit DDR input data, and optionally a synchronization signal, synchronous to a RXCLK signal. It then uses this same clock to deserialize the data from 16 bits DDR input to 128 bits wide internally. The 128-bit wide data is then written into a FIFO using four block memories.

The output of the FIFO together with the necessary control signals are available to the designer for passing data further into the system. The input timing characteristics of the 16-bit block are as shown for the 4-bit block. The block has been designed to be used on the left-hand edge of the Virtex-II device, but could also be placed on the right if necessary. The placement of the individual modules is defined by the rloc_origin statement applied to each module in the constraints (design.ucf) file used when processing the design through Xilinx implementation tools. These locations vary with device size and pinout chosen, but a typical example for the XC2V1000 device would be:

```
'set "rx0/hset" rloc_origin = "X0Y0" ;'
'set "rx1/hset" rloc_origin = "X0Y8" ;'
'set "rx2/hset" rloc_origin = "X0Y16" ;'
'set "rx3/hset" rloc_origin = "X0Y24" ;'
```

Each 4-bit macro is four CLBs high by three CLBs wide to have the same pitch vertically as the block RAMS. The four macros necessary for 16-bit use can be placed immediately adjacent to each other vertically, or they can be separated as shown in Figure 10. Careful consideration also has to be given to the pins chosen for the receiver input. Ideally, they will be immediately adjacent to the macro, but a certain amount of spread is acceptable, plus or minus two CLBs vertically still works at up to 840 Mb/s in a -5 device.

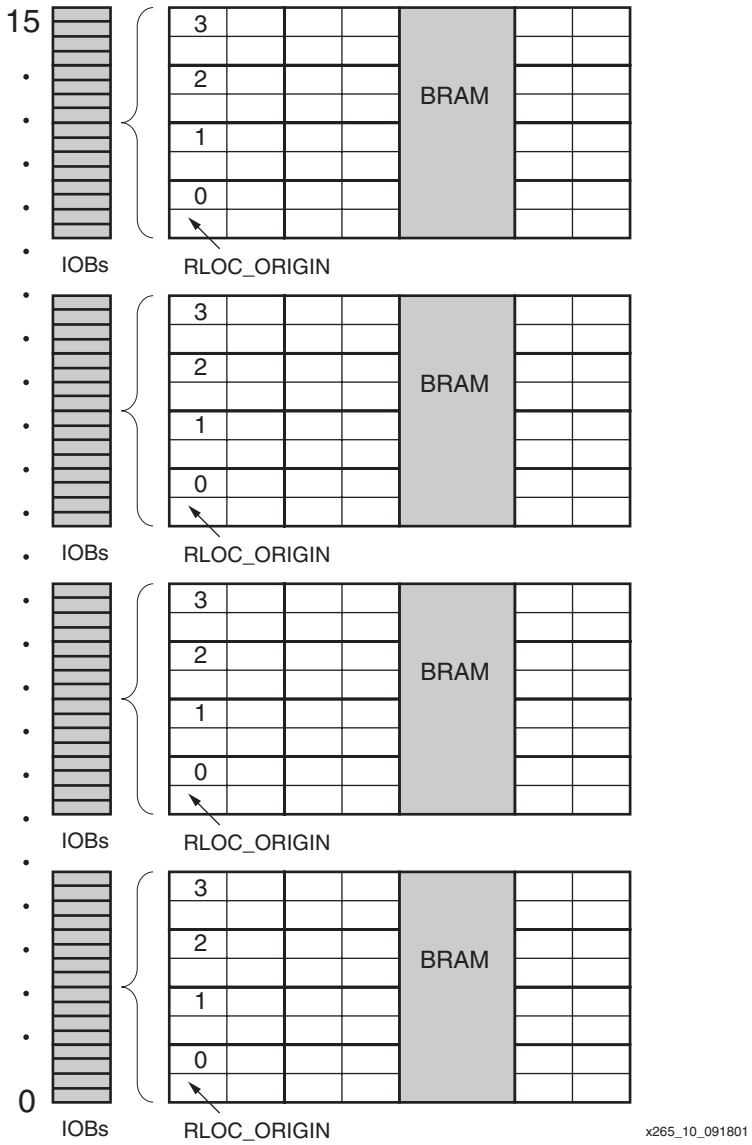


*Figure 10:* **16-Bit Receiver Macro Physical Implementation**

## DCM operation

As mentioned, the receiver clock is derived from the forwarded clock via a DCM which has a constant or variable phase shift programmed into it. The method for doing this is to add attributes for the phase shift mode (clkout_phase_shift) and for the value of phase shift (phase_shift) to the DCM instantiation as shown in top16.vhd. The goal of the phase shift is to place the middle of the data "eye" into the middle of the setup window of the input flip-flops. The value for this phase offset is calculated as follows.

Assuming that the clock and data arrive at the receiver device at the same time, the required time shift is one quarter of the incoming clock period plus the mid-point between the pin-to-pin setup time and the pin-to-pin hold time for IOB flip-flops given in the Virtex-II data sheet. Note that it is not necessary to add the correction factor for LVDS as both the clock and data are the same standard, and so will be equally delayed in the IOB. The required phase shift value will then be the total time shift calculated divided by the clock period and multiplied by 256.

For example, if we are using 622 Mb/s signalling (311 MHz clock), the period will be 3.215 ns, and a quarter period will be 0.803 ns. If the pin-to-pin setup is 1.5 ns, and the pin-to-pin

hold is –0.9 ns, then the mid-point is 1.2 ns. By adding 0.803 and 1.200 the resultant is 2.003 ns. Dividing by the clock period gives 0.623, and multiplying by 256 produces a value of 160. Thus, for correct operation of the circuit, the DCM will need a value of 160 applied to it.

## Reference Design

The VHDL code for the implementations discuss in this application note are available on the Xilinx FTP site at:

ftp://ftp.xilinx.com/pub/applications/xapp/xapp265.zip

The readme.txt file includes further implementation details.

## Conclusion

The Virtex-II device in a -5 speed grade can be shown to be capable of 16-bit LVDS data transmission and reception with or without a framing signal, at up to 840 Mb/s using DDR techniques.

# Appendix A

## 7:1 and 1:7 SerDes Modules

As an alternative to certain popular dedicated LVDS devices, Virtex-II devices can also be used in systems requiring a SerDes factor of seven. Typically, the forwarded clock does not run at the line rate, but at the line rate divided by seven. This allows the forwarded clock to serve as a framing signal, as well as a clock signal. Using the clock multiplier built into the Virtex-II DCM, transmission rates at up to 640 Mb/s per pin-pair are possible with a -5 device.

### Transmitter

The transmitter design for a 7:1 SerDes is similar to the 8:1 SerDes design (**Transmitter Module**, page 2). In the 7:1 SerDes, the two required clocks are a x1 clock, and a x3.5 clock. DDR data is transmitted by the x3.5 clock, and the forwarded clock is generated by the x1 clock (Figure 11). Using the digital frequency synthesizer in the DCM blocks, the two clocks are generated. The digital frequency synthesizer has a maximum output frequency of 320 MHz, and therefore a maximum input frequency of 320 divided by 3.5 (91 MHz in this application). Data framing is automatically processed, as the first nibble transmitted is always synchronized to the forwarded clock.

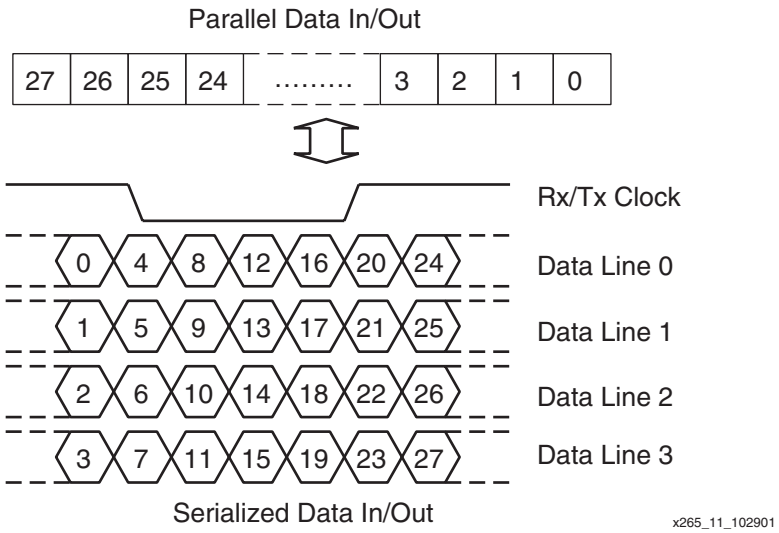


*Figure 11:* **Data Formatting**

The basic transmitter module (VHDL and Verilog) is in the 7:1 subdirectory of the reference design file **xapp265.zip** available on the Xilinx FTP site. The module takes in 28 bits of data, and serializes it to four lines running DDR at 3.5 times the clock rate. The bit ordering shown in Figure 11, is easily changed with a design changing the bit order between the transmitting system and the macro. The reference design files include specific examples. Some systems only transmit 24 bits and use the "spare" bit positions for a DC balancing function on all four pin-pairs of the transmitter. An example of this is shown in the reference design file serdes_4b_7to1_wrapper_483. The balancing bits are generated (via LUT ROMs) using a very simple algorithm. If there are more "ones" than "zeros" in the 6-bit serial transmission, then an extra zero is used for the seventh bit. If there are more zeros than ones, then an extra one is used for the seventh bit. This system helps ensure that the transmitted line is close to being DC balanced, and helps the PCBs signal integrity.

The placement constraints for the transmitter are the same as for the 8:1 SerDes macro. Additionally, if required the 4-bit macro can be cascaded for 8-bit links (or wider). An example of an 8-bit transmitter is shown in the (serdes_8b_7to1) file in the reference design.

### Receiver

The 1:7 receiver module is also very similar to the 1:8 module described in **Receiver Module**, page 7. Four bits of data are received and deserialized to 28 bits of data, available to the

system via a shallow FIFO. The incoming clock is used both for receiving data, and for framing data into the correct bit positions (as shown in Figure 11). The code for a basic 1:7 receiver is given in reference design (serdes_4b_1to7).

In a Virtex-II DCM, multiply the incoming clock by 3.5, then phase shift the clock in the incoming data. The method for calculating the required phase shift is as given in the main body of the application note. The clock is also used by the receiver module for correctly aligning the bits in the receive FIFO. The bits will be received in the same pattern as they are presented.

The received bit order can be modified when using the reference design files. The placement constraints are very similar to those for the 1:8 macro. Multiple receivers can be cascaded for use with connections of eight or more pin-pairs.

## Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 10/04/01 | 1.0 | Initial Xilinx release. |
| 11/07/01 | 1.1 | Added Appendix A. |