



XAPP460 (v1.1) June 24, 2011

# Video Connectivity Using TMDS I/O in Spartan-3A FPGAs

Author: Bob Feng and Eric Crabill

## Summary

Transition Minimized Differential Signaling (TMDS) is a standard used for transmitting video data over the Digital Visual Interface (DVI) and High-Definition Multimedia Interface (HDMI). Both interfaces are commonly used by consumer electronics including flat panel TVs, PC monitors, DVD players, PC desktops, and video game consoles.

This application note describes a set of reference designs able to transmit and receive DVI or HDMI data streams up to 700 Mb/s using the native TMDS I/O interface featured by Spartan®-3A FPGAs.

## Introduction

The DVI and HDMI protocols use TMDS at the physical level. TMDS throughput is a function of the serial data rate of the video screen mode being transmitted. This in turn determines the Spartan-3A FPGA speed grade that must be used to support this throughput.

### FPGA Throughput and Video Screen Modes

Spartan-3A FPGAs feature native TMDS I/O with compliant differential voltage standards. [Table 1](#) shows the maximum throughput for each Spartan-3A FPGA speed grade.

**Table 1: Spartan-3A Family TMDS I/O Throughput**

Speed Grade	Throughput
-5	700 Mb/s
-4	640 Mb/s

Common video screen modes corresponding to these data rates are listed in [Table 2](#).

**Table 2: Common Video Screen Modes**

Screen Mode	Pixel Rate	Serial Data Rate	Color Depth
VGA (640x480@60 Hz)	25 MHz	250 Mb/s	24 bits
480p <sup>(1)</sup> (720x480@60 Hz)	27 MHz	270 Mb/s	24 bits
SVGA (800x600@60 Hz)	50 MHz	500 Mb/s	24 bits
XGA (1024x768@60 Hz)	65 MHz	650 Mb/s	24 bits
HDTV 720p <sup>(1)</sup> (1280x720@60 Hz)	74.25 MHz	742.5 Mb/s	24 bits
HDTV 1080i <sup>(1)</sup> (1920x1080@60 Hz interlaced)	74.25 MHz	742.5 Mb/s	24 bits

#### Notes:

1. Per CEA-861-D [\[Ref 1\]](#).

## TMDS Video Transmission Protocols

This section briefly discusses the TMDS video transmission protocols. Detailed information is available in these documents:

- *A DTV Profile for Uncompressed High Speed Digital Interfaces* [Ref 1]
- *Digital Visual Interface, Revision 1.0* [Ref 2]
- *High-Definition Multimedia Interface Specification, Version 1.3a* [Ref 3]

Figure 1 shows the topology for an established TMDS link between the transmitter (Source) and receiver (Sink).

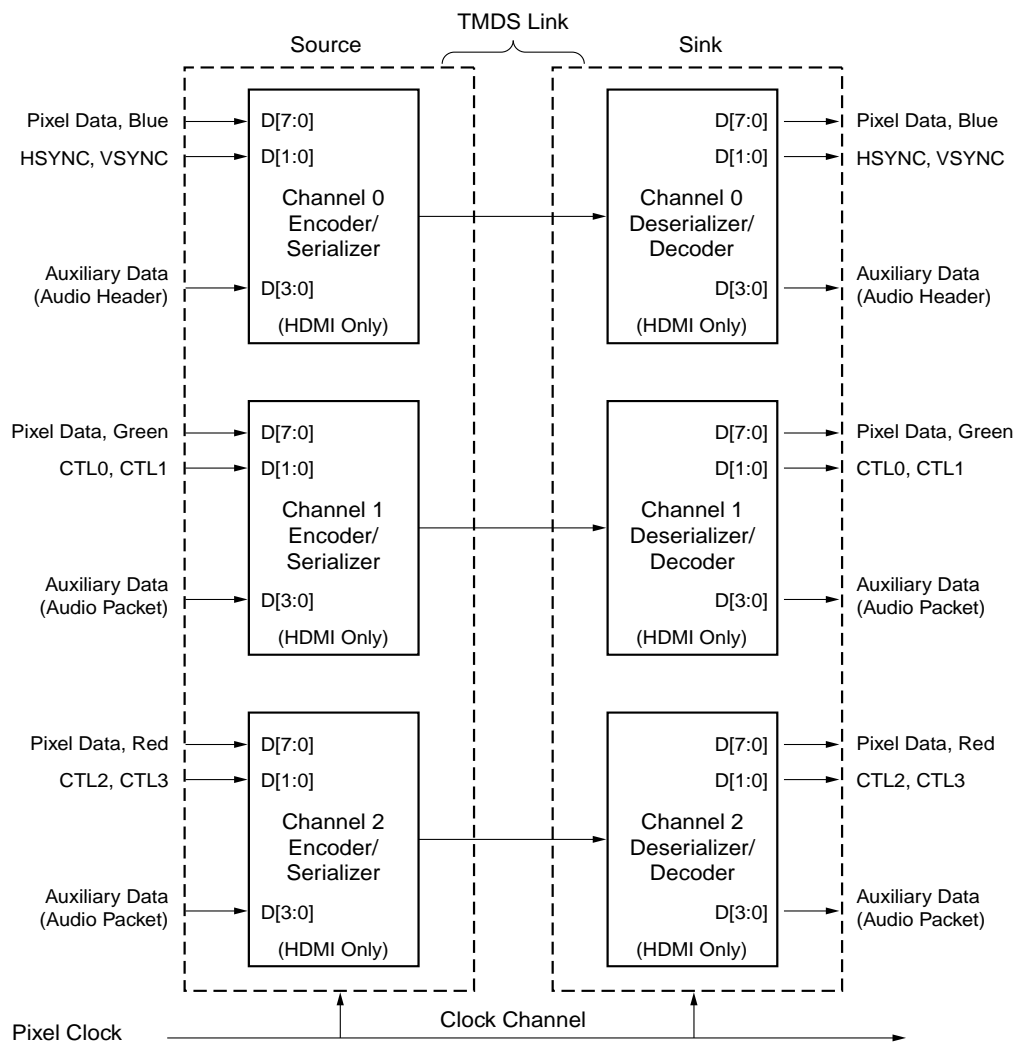


Figure 1: TMDS Link Topology

Four channels of serial data establish a single link of DVI or HDMI video transmission. In DVI, three channels are designated for the red, green, and blue color components (RGB 4:4:4) in each video pixel data. HDMI also uses three RGB channels by default, but they also can be optionally used to carry Luminance and Chrominance components (YCrCb 4:4:4 and YCrCb 4:2:2). The fourth channel transmits a clock toggling at the pixel data rate.

For a pixel having 24-bit color depth, each color component originates as 8-bit data, which is then converted into a 10-bit symbol using an 8B/10B encoding scheme. The 10-bit symbol is then serialized and transmitted onto one of the TMDS data channels. This 10:1 serialization ratio results in a 10x faster bit rate than the actual pixel rate.

During the video frame transmission, the active pixel symbol is periodically interlaced with four distinct control tokens representing blanking intervals. These control tokens provide accurate video line scan (HSYNC) and frame update (VSYNC) information. Control tokens are also used to identify word boundaries for synchronization purposes.

For DVI transmission, the full video blanking period is used only to transmit control tokens. An HDMI transmission also defines a Data Island in the blanking period to transmit packets of audio data and auxiliary data. The auxiliary data includes InfoFrames and other descriptive data. This is the main difference between the two protocols.

## Logic Construct

This section describes the use of the Spartan-3A family's built-in resources to construct a soft DVI/HDMI transceiver. The transmitter and the receiver designs are covered separately.

### Transmitter Design

Video transmission over the TMDS link is logically divided into an encoder and serializer stage as illustrated in Figure 2. The encoders convert pixel data from a video source, HDMI Auxiliary/Audio data, and HSYNC and VSYNC into three 10-bit symbol streams. The serializer performs a parallel-to-serial conversion on all three streams and then sends them out onto three channels of differential output pairs. The serialization ratio is 10:1 per channel.

Figure 2 also shows how the transmitter is physically implemented, with the design of the encoder and serializer logic located in the FPGA logic while the dedicated I/O Block (IOB) contains the ODDR2 double data rate output registers and the TMDS differential output buffers.

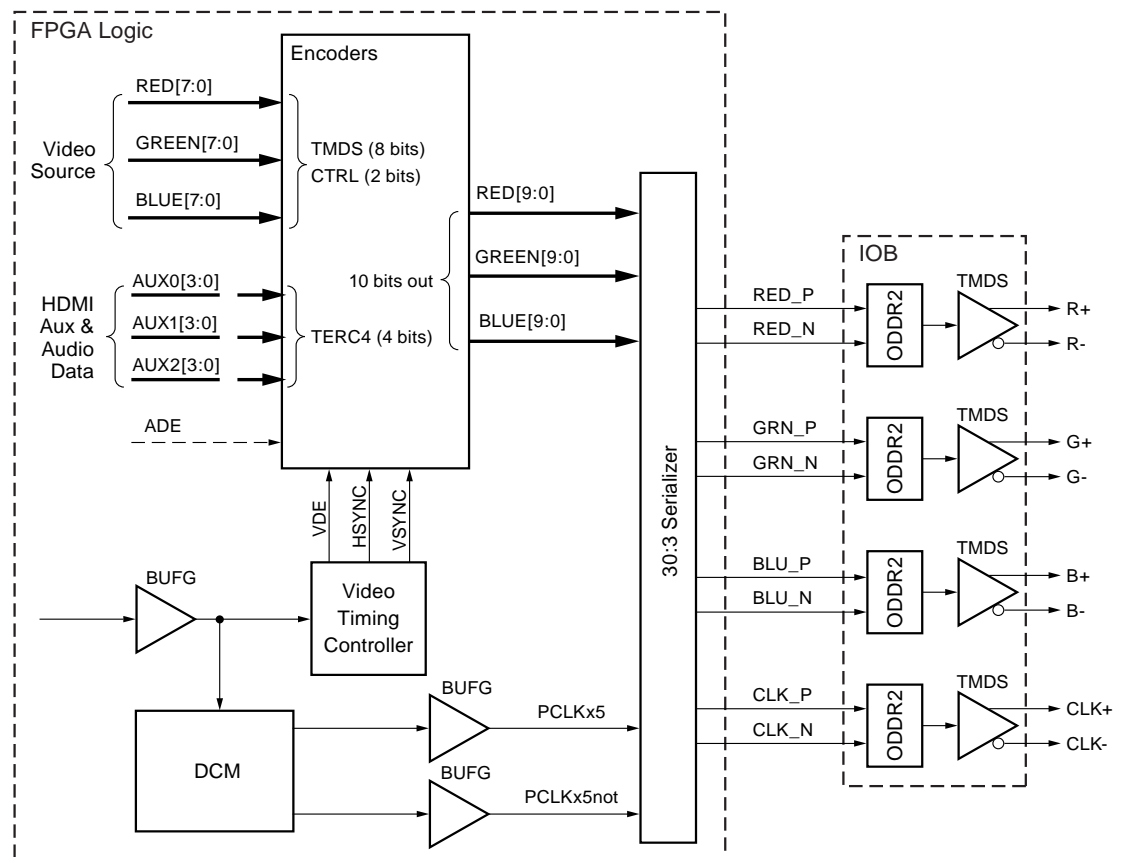


Figure 2: TMDS Transmitter

X460\_02\_070908

### DVI Encoder

The DVI encoder outputs pixel data during the active video period and control data during the blanking period. Figure 3 shows the relationship of Video Data Enable (VDE) to these two states. VDE, HSYNC, and VSYNC are generated by the Video Timing Controller shown in Figure 2.

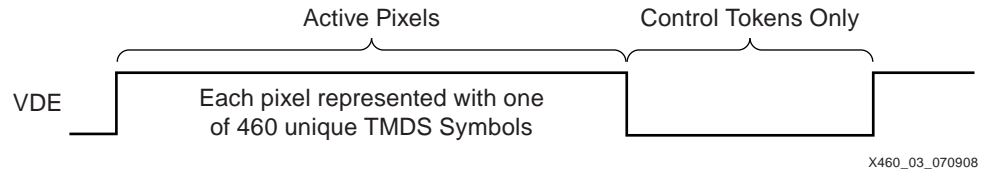


Figure 3: DVI Encoding Scheme

Figure 4 shows the encoder layout for all three channels. The encoder logic is identical for all three pixel components. The state of VDE defines the active video and the blanking periods and enables the output of either video pixel data or control data. HSYNC and VSYNC are encoded on the blue channel for transmission during the blanking period. Control signals C0 and C1 are encoded and output during the blanking period on the green and red channels. The DVI specification reserves the use of control symbols C0 and C1, so these are a value of 2'b00 at the input of each encoder.

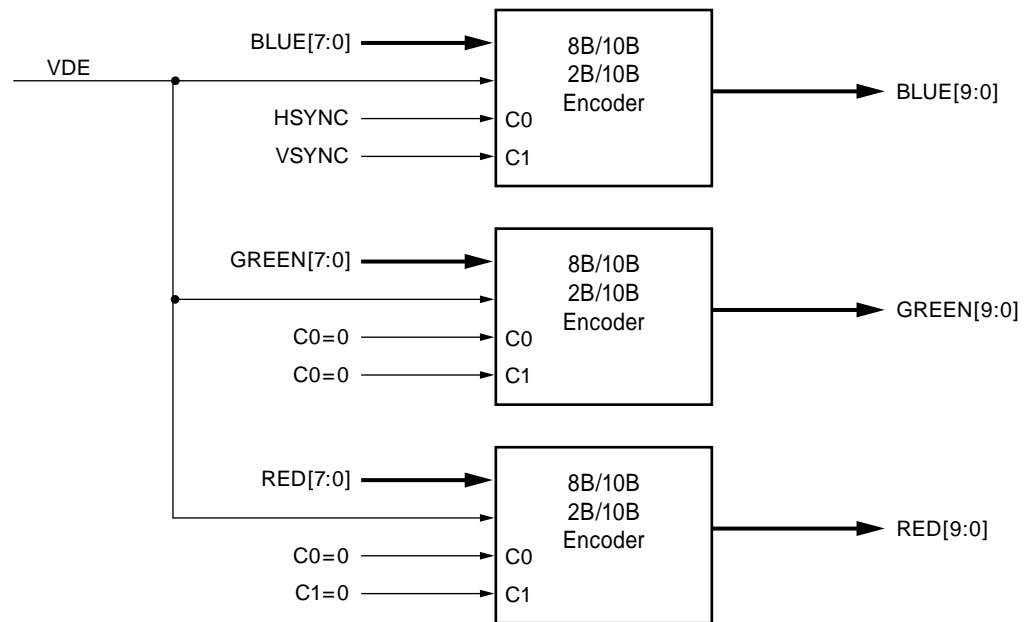


Figure 4: DVI Encoder Layout

The incoming video pixel values for each channel are encoded using the TMDS encoding algorithm defined in the DVI specification. Each 8-bit value is converted into one of 460 unique 10-bit symbols. This encoding scheme serves to achieve an approximate DC balance and limit the number of state transitions to five or fewer per symbol.

The status of the 2-bit control bus for each channel is transmitted during the video blanking period. This data is represented by four distinct 10-bit control tokens predefined as: 10'b1101010100, 10'b0010101011, 10'b0101010100, and 10'b1010101011. Each token has seven or more state transitions. The difference in the number of state transitions between video symbols and control tokens will be used to synchronize the transmitting and receiving devices.

## HDMI Encoder

The HDMI protocol is derived from and shares many aspects with the DVI protocol, including the physical TMDS link, the active video encoding algorithm, and the control token definitions. HDMI carries much more data than DVI by transmitting auxiliary data (InfoFrames) and audio data during the video blanking period. Figure 5 shows the HDMI encoding scheme for video and audio data relative to VDE and Aux/Audio Data Enable (ADE).

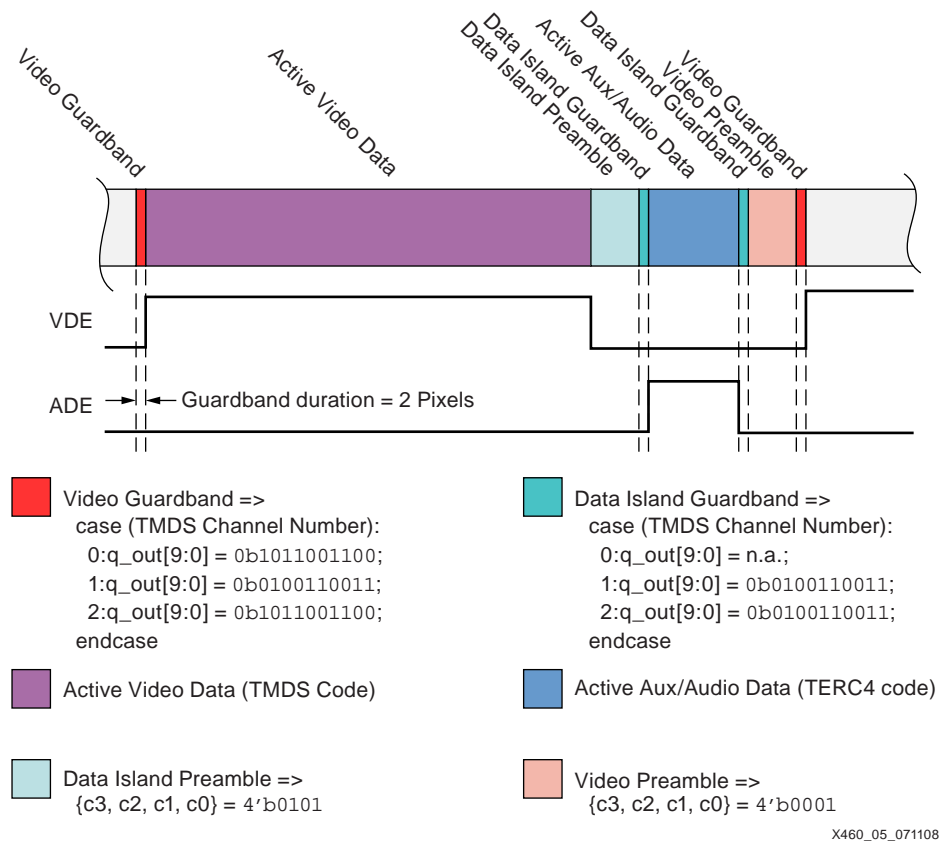


Figure 5: HDMI Encoding Scheme

The HDMI encoder converts each 8-bit pixel component into one of 460 unique 10-bit symbols using an encoding algorithm similar to the one used by the DVI specification. A representative flowchart describing this algorithm is shown in the HDMI 1.3 specification.

The information contained in each segment of the HDMI data stream is described in the following paragraphs.

**Active Video Data.** Video pixels are encoded using the same algorithm used by DVI. Each 8-bit pixel component is converted into one of the 460 unique 10-bit symbols. A similar process flow chart is illustrated on page 82 in HDMI specification, version 1.3.

**Data Island Preamble.** An advanced indication that the Data Island period (containing audio samples and InfoFrames) will follow. Contains eight identical, consecutive control tokens on the green and red channels only.

**Data Island Guardband (Leading).** Provides design with a means of synchronization between transmitter and receiver. Guardband has a 2 pixel duration.

**Active Aux/Audio Data.** Auxiliary data (InfoFrame) and audio data encoded as 10-bit TERC4 symbols. The TERC4 encoding scheme consists of 16 unique 10-bit characters converted from 4-bit auxiliary or audio data. Transmitted on the green and red channels only.

**Data Island Guardband (Trailing).** Indicates the end of the Data Island and has a two-pixel duration. Transmitted on the green and red channels only.

**Video Preamble.** An advanced indication that the Active Video period will follow. Contains eight identical, consecutive control tokens on the green and red channels only.

**Video Guardband (Leading).** An advanced indication that the Active Video Data period will follow.

Control Tokens are also managed by the HDMI encoder and:

- Indicate the start of the blanking period in the data island preamble
- Indicate control bus {C1, C0} status
- Provide HSYNC and VSYNC on the blue channel
- Output reserved tokens for {C1, C0} = 2'b00 on the green and red channels
- Output Video and Data Island Preambles on the green and red channels

Figure 6 shows the HDMI encoder layout. Similar to the DVI encoder (Figure 4), the video timing signals HSYNC and VSYNC are transmitted on the blue channel (channel 0). HDMI encoding differs from DVI in that the other two channels are assigned to transmit the preamble control tokens.

Before the blanking period transits into the active video period, at least eight video preambles followed by two video starting guardbands have to be transmitted ahead. Similarly before a frame of auxiliary data starts, exactly eight consecutive data island preambles followed by two Data Island starting guardbands have to be transmitted first.

To transmit the video and data island preambles at the correct time, the video/blanking and blanking/auxiliary data transitions need to be predetected. This can be done by creating a certain amount of delay between the video/audio data and the control buses as shown in Figure 6.

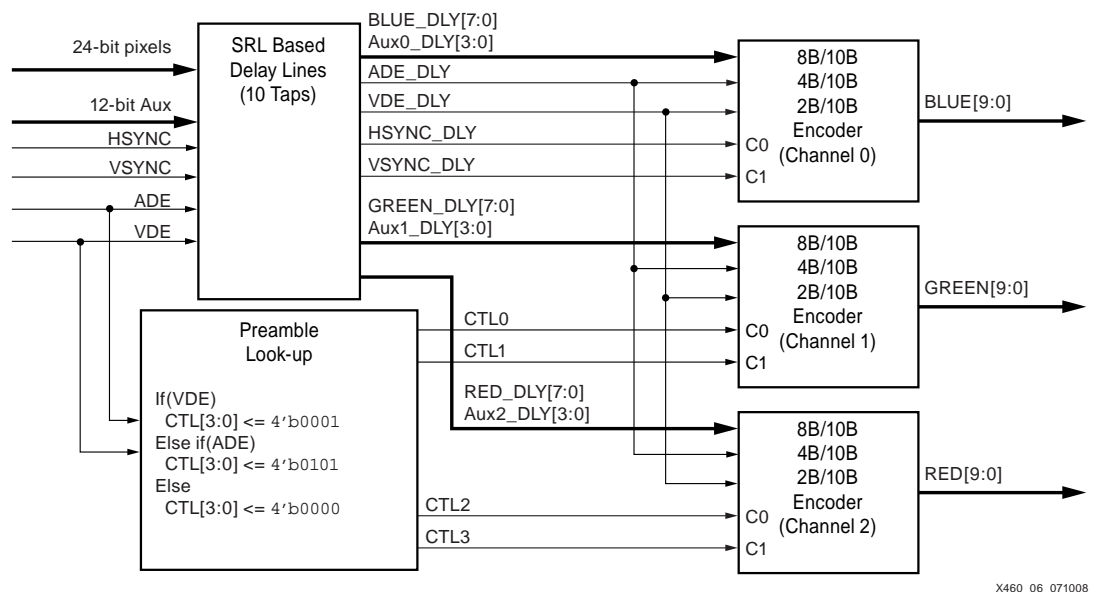


Figure 6: HDMI Encoder Layout

The gaps are achieved using SRL16 based delay lines for all 40 bits of the video and auxiliary signals. In this design, the SRL16 is configured to provide 10 taps delay. This accommodates the required eight consecutive preambles plus two cycles of internal pipeline delay in these encoders.

By postponing the video and auxiliary signals fed into the 8B/4B/2B to 10B encoder, the preambles look-up table controlled by the non-delayed version of VDE and ADE provides the appropriate predetection of the video/blank/auxiliary transitions.

Figure 7 shows a simulation of the preamble data generation on the green channel. With the right amount of delay between the transitions of the CTL[3:0] bus and VDE\_DLY/ADE\_DLY, exactly eight consecutive data island preambles are transmitted before the data island guardbands containing the auxiliary TERC4 symbols.

The HDMI encoder transmits HSYNC and VSYNC through the blue channel. These signals are transmitted for the entire video blanking period including the data island and the preamble periods. During the data island period these signals are encoded using the TERC4 scheme together with two MSB bits from Aux0. Therefore only Aux0[3:2] are valid for the blue channel. The HDMI specification defines these two bits as sampled audio and InfoFrame headers.

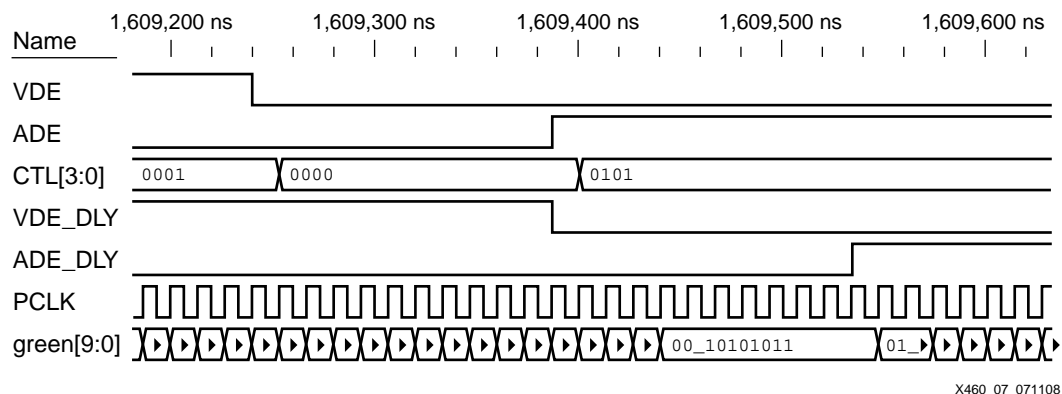
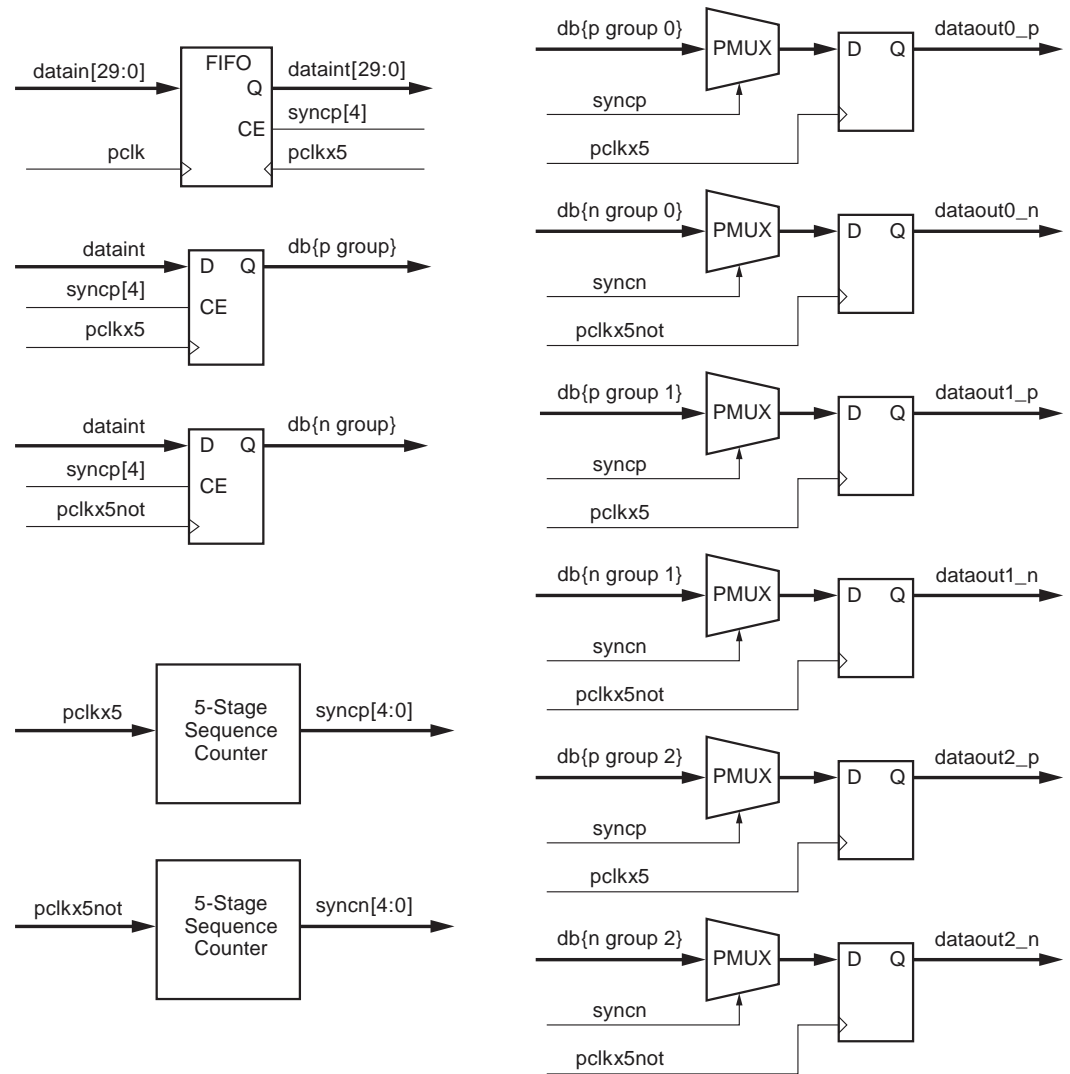


Figure 7: Simulation of Preamble Generation for the Green Channel

### Serializer and Clocking Schemes

The DVI and HDMI transmitters share identical serializer designs and clocking schemes.

After the encoding process, all three 10-bit symbols are grouped together and fed into a 30:3 serializer (Figure 8). Its 10:1 serialization ratio represents a bit rate that is 10 times the pixel rate. When transmitting an HDTV 720p screen mode video stream, the serial data rate is 742.5 Mb/s. The only practicable means to accomplish conversions at this rate is to apply a Double Data Rate (DDR) technique inside the FPGA. This reduces the bit sampling clock by half to 371.25 MHz.



X460\_08\_062008

Figure 8: 30:3 Serializer

The Spartan-3A family provides up to eight Digital Clock Manager (DCM) blocks. In this application, a DCM is used to synthesize the pixel clock (pclk) multiplication. The DCM synthesizes the pclkx5 and pclkx5not clock. pclkx5not is phase-shifted 180° relative to pclkx5. These clocks drive the Stage sequencers that determine bit positions and the bit select multiplexers that shift the bit out. The basic building blocks of the serializer are shown in Figure 8.

An asynchronous FIFO and a set of latch registers are used to transfer the 10-bit encoded symbols from the pclk domain into the pclkx5 and pclkx5not domains. As a result, there is no phase alignment requirement between pclk and pckx5. This approach is highly recommended to avoid any bit errors due to alignment skew at high frequency, such as the 371.25 MHz here.

The FIFO can be constructed using either the FPGA internal block RAMs or the unique Xilinx distributed RAMs by leveraging the look-up tables in the fabric. The FIFO is only used as a data bridge between the two clock domains. Average reads and writes are guaranteed to have the same speed by using clock enables. Thus, the depth requirement is shallow and a complex flow control mechanism is not needed, permitting the FIFO to free run.

A soft serializer design is often required to be bound together with the hardened IOB resources. By using distributed RAMs, this requirement can be relaxed, permitting more flexibility with

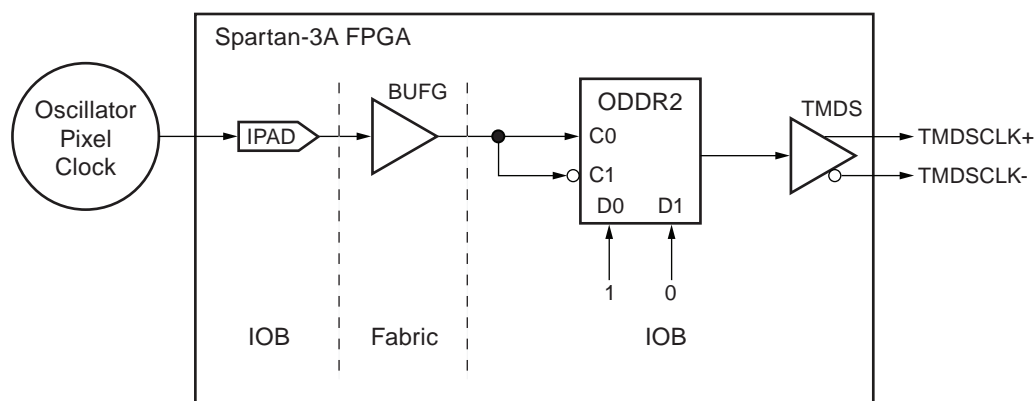


placement of the serializer within the FPGA. While the block RAM offers much deeper depth, its location inside the FPGA is fixed. It cannot provide the same location flexibility as distributed RAM. Moreover, the placement of the block RAM with respect to the IOB can create timing barriers if the selected IOB is far away. For this application, using distributed RAM to construct the FIFOs is recommended.

Other than the pixel or auxiliary data itself, a DVI/HDMI transmitter is specified to forward the pixel rate clock through an additional differential pair. Unlike other differential signaling standards such as LVDS, the DVI and HDMI specifications do not have any phase alignment requirement between the TMDS clock and the TMDS data. The only requirements are:

- Pixel rate matching frequency
- Period jitter  $\leq 0.25$  UI [Ref 2], [Ref 3] (unit interval or bit time)

Although there are many ways to generate the forwarding clock, Xilinx recommends using the scheme shown in Figure 9. The pixel clock is routed from an external source (such as an oscillator) into the FPGA through an input pad (IPAD), global clock buffer (BUFG), and a clock network to drive an IOB ODDR2 register directly. The ODDR2 C0 input is connected to the BUFG output. The C1 input is driven through a local inversion inside the IOB. The D0 and D1 inputs are connected to logic 1 and logic 0, respectively. This guarantees the generated forwarding clock has the minimum amount of accumulated jitter by minimizing the number of logic components in the path to the TMDS output buffer.



X460\_09\_062008

Figure 9: TMDS Clock

## Floorplanning

The 30:3 serializer needs to operate at five times the pclk frequency, or 371.25 MHz, for the HDTV mode. Because it also directly drives the ODDR2 registers inside the IOB blocks, it is important to group the serializer components close together and locate this group as close as possible to the IOBs. Figure 10 shows the serializer floorplan grouped this way.



X460\_10\_070208

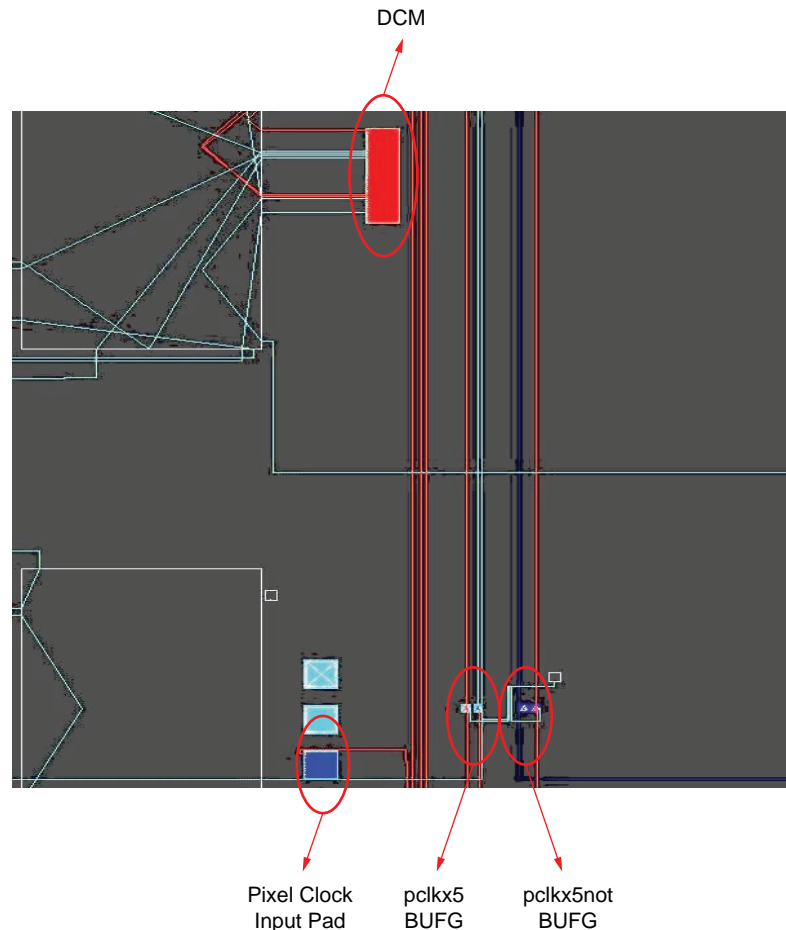
Figure 10: **30:3 Serializer Floorplan**

The serializer components can be grouped and placed in a rectangular area with a user constraint file (UCF) using statements:

```
INST "dvi_tx0/serialise/*" AREA_GROUP = "AG_serializer";
AREA_GROUP "AG_serializer" RANGE = SLICE_X4Y143:SLICE_X15Y136;
```

The serializer with 12x8 slices is gathered into the AG\_serializer group and placed near the relevant IOBs.

Another aspect to consider is the placement of the DCM and BUFGs. Both BUFGs for pclkx5 and pclkx5not should be placed on the same side as their source DCM, for example, on bank 0 as shown in [Figure 11](#).



X460\_11\_061908

Figure 11: Transmitter DCM and BUFG Placements

The corresponding UCF statements for the transmitter DCM and BUFG placements are:

```
INST "DCM_SP_INST" LOC = "DCM_X1Y0";
INST "pclkx5notbufg" LOC = "BUFGMUX_X1Y0";
INST "pclkx5bufg" LOC = "BUFGMUX_X2Y1";
```

Using these statements will achieve the minimum skew between the two clocks.

### Transmitter Timing Analysis

There are three clock domains in the transmitter design: pclk, pclkx5, and pclkx5not. Everything in the transmitter except the 30:3 serializer runs at the pclk frequency. The UCF should constrain pclk as shown here:

```
NET "clk" TNM_NET = "TNM_NET_PCLK";
TIMESPEC "TS_PCLK" = PERIOD "TNM_NET_PCLK" 74.25 MHz HIGH 50% PRIORITY 0;
```

The Xilinx ISE® timing analysis tool can trace pclk through the DCM as well as the BUFGs and then automatically add pclkx5 and pclkx5not constraints. Thus, there is no need to explicitly list the frequency limits of pclkx5 and pclkx5not in the UCF.

Timing on these FIFO paths must be carefully considered:

- dataint outputs to db at the latching registers relative to pclkx5 and pclkx5not
- Read address ports ra to db at the latching registers relative to pclkx5 and pclkx5not

These are multicycle paths since they are clock enabled through the syncp[4] or syncn[4] pulses, which toggle once only every five pclkx5 or pclkx5not clock cycles.

For paths that are latched by pclkx5, the frequency constraint only needs to be set to pclkx5/5.

For paths that are latched by pclkx5not, it is necessary to set the constraint to pclkx5/4.5 because pclkx5 and pclkx5not are 180 degrees out of phase.

For this application, all the FIFO paths are constrained to pclkx5/4 as shown in this example:

```
#####
# MultiCycle Paths
#
# Although these flip-flops are in clkx5 and clkx5not domains, they are
# clock enabled through sync pulses that toggle only every five clock
# cycles.
#
# Because the sync pulse to clkx5not timing is set to occur one half
# clock earlier than the pulse to clkx5, the timing is then 4.5 cycles
# of clkx5not.
#
# Set the timing to meet 74.25 * 5/4.
#
#####
TIMEGRP "bramgrp" = RAMS(dvi_tx0/serialise/dataint<*>);
TIMEGRP "fddbgrp" = FFS(dvi_tx0/serialise/db<*>);
TIMESPEC "TS_ramdo" = FROM "bramgrp" TO "fddbgrp" TS_PCLK * 1.25;

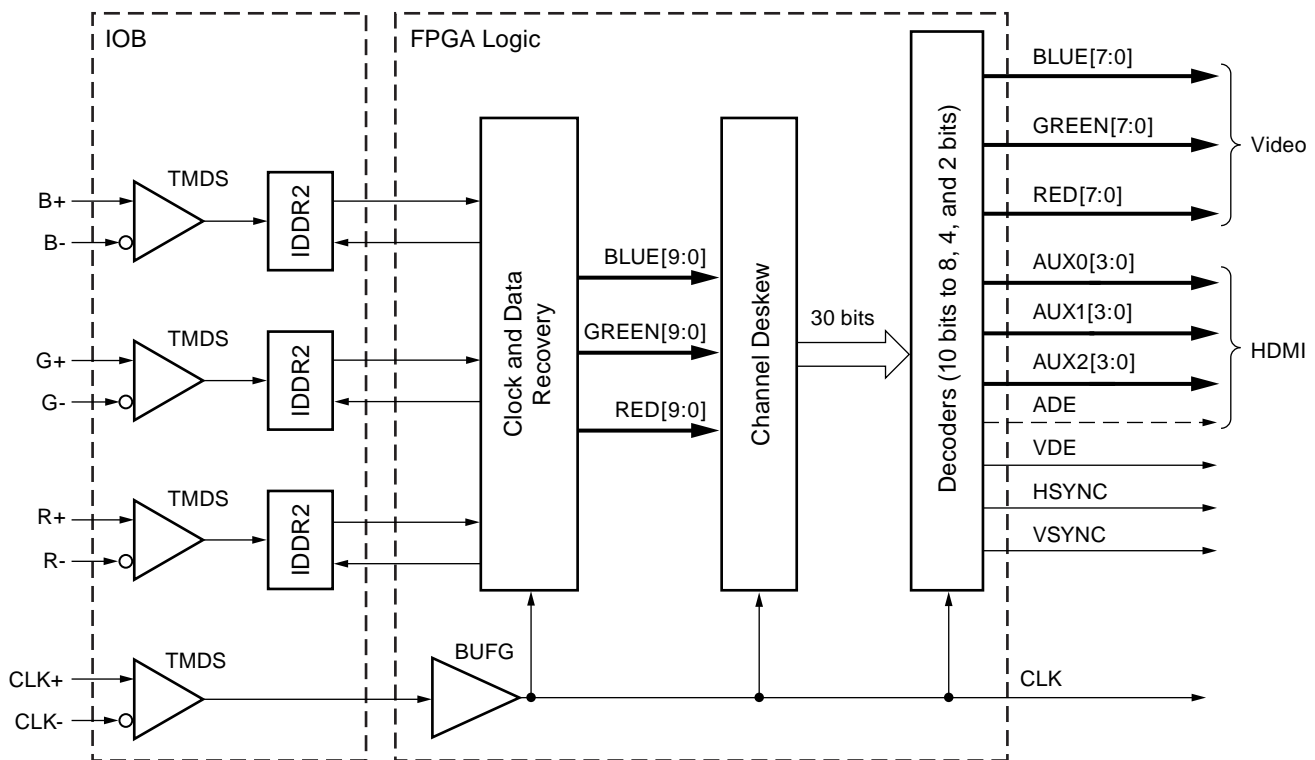
TIMEGRP "bramra" = FFS(dvi_tx0/serialise/ra<*>);
TIMESPEC "TS_ramra" = FROM "bramra" TO "fddbgrp" TS_PCLK * 1.25;
```

## Receiver Design

The receiver needs to recover the bit sampling clock using the incoming pixel clock, and then apply this bit sampling clock to recover the serial data stream back into 10-bit word aligned symbols. This process is known as clock and data recovery (CDR). After this step, a channel deskew circuit removes allowed skews among the three data channels. Finally the 10-bit symbol is converted back to its original form containing:

- 8-bit video pixel data (recovered using either the DVI or HDMI decoder)
- 4-bit auxiliary data (InfoFrames and sampled audio data recovered through the HDMI decoder)
- 2-bit control data (HSYNC and VSYNC recovered using either the DVI or HDMI decoder)

Figure 12 illustrates the receiver data process flow.



X460\_12\_062008

Figure 12: TMSD Receiver

### Clock and Data Recovery

The TMSD clock channel input provides a reference clock running at the character rate frequency. The receiver uses this clock to sample the incoming serial streams. However, this sampling clock does not have a guaranteed phase relationship associated with any of the three data channels. Consequently, the phase between this clock and the serial data on each channel must be adjusted to correctly sample each incoming bit.

By definition, the bit time or Unit Interval (UI) is only 1/10th of a pixel period, and is equivalent to the pixel rate frequency defined for each video screen mode (Table 2). For 720p and 1080i high-definition video, the pixel clock frequency is 74.25 MHz. The corresponding serial data rate is 742.5 Mb/s, and the UI is approximately 1.3 ns.

In a real digital transmission system, jitter describes the arrival time uncertainty on transition edges of clock and data. Jitter reduces the margin for obtaining a valid sample. The window that defines this margin is described as an eye. The width of the eye is always shorter than the UI. To ensure a valid sample, the phase of the sampling clock must be adjusted so the sample occurs near the middle of this eye.

Skews can also exist between all three data channels. The DVI and HDMI specifications require the compliant receiver to tolerate the skew up to  $0.6 \times T_{\text{pixel}}$ .

Thus, the key to recovering video data from a TMSD transmitter is to adjust the reproduced bit clock into the middle of the data eye on a per channel basis. After data is recovered, the receiver must also remove skews between any two of three data channels.

There are many ways to conduct asynchronous clock to data recovery. Two examples using oversampling can be found in Xilinx application notes XAPP224 [Ref 4] and XAPP861 [Ref 5].

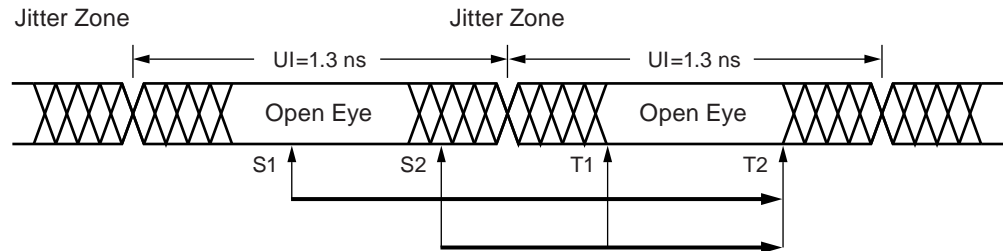
The method used by this application uses the DCM's variable phase shift capability to dynamically adjust the clock into the middle of the data eye of each channel. It also applies the DDR technique to permit the use of a 5X sampling clock instead of a 10X sampling clock. This

makes it possible to implement video connectivity in a low-cost device, like a Spartan-3A FPGA.

The first challenge to recovering the transmitted data is how to locate the open eye when the phase relationship between the sample clock and data is unknown.

This uncertainty yields only two possibilities:

- The clock is positioned in the data's open eye (S1 in Figure 13)
- The clock is positioned in the data's jitter zone (S2 in Figure 13)



X460\_13\_070208

Figure 13: Two Unit Intervals of the TMDS HD Video

The video data transmitted over the TMDS channels is essentially random, but the four distinct control tokens are periodically transmitted and can be used to detect the position of the clock relative to the data. The DVI specification ensures the minimum blanking period is at least 128 pixels for the transmission of the control tokens, but the HDMI specification reduces the control token period to 12 pixels while allowing the rest of blanking period to transmit audio or auxiliary data. Thus, the detection method must recognize the occurrence of either 128 or 12 control tokens from the transmitter. If the control tokens are detectable within a time limit, the clock is assumed to have arrived in the open eye position. Otherwise, the clock is assumed to have arrived in the jitter zone. While shifting the clock phase, the beginning positions of the open eye (T1) and jitter zone (T2) are recorded as shown in Figure 13. Therefore, the middle of the open eye is calculated to be  $(T2 - T1)/2$ .

Below is the detailed process:

If the clock initially arrives at marker position S2:

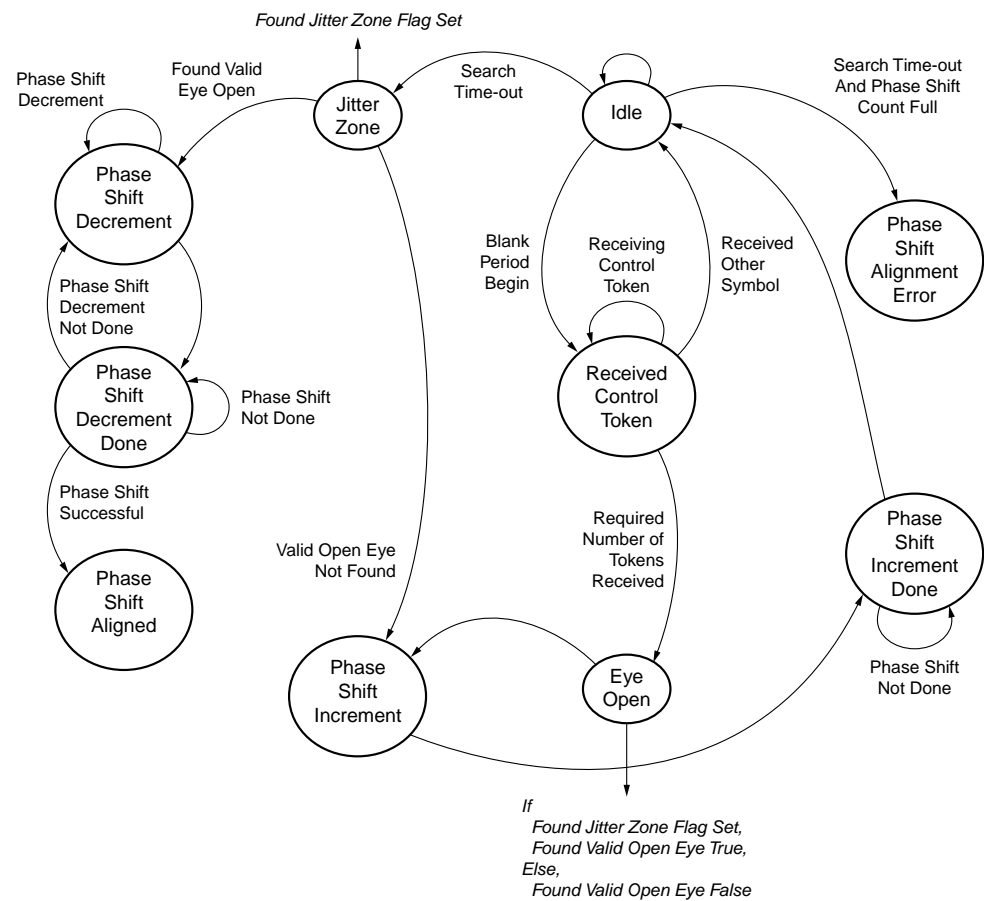
- The phase is incremented toward T1.
- T1 is recorded.
- The phase is continuously incremented until T2.
- T2 is recorded.
- The phase is decremented back to  $(T2 - T1)/2$ .

If the clock initially arrives at marker position S1:

- The current open eye is abandoned and the next eye is used.
- The phase is incremented toward S2 where the next jitter zone starts.
- The phase is incremented until T1.
- T1 is recorded.
- The phase is continuously incremented until T2.
- T2 is recorded.
- The phase is decremented back to  $(T2 - T1)/2$ .

The algorithm requires two UIs for the phase shift to handle both cases discussed above. The phase shift step resolution affects the precision of the algorithm, and the maximum phase shift range determines its success.

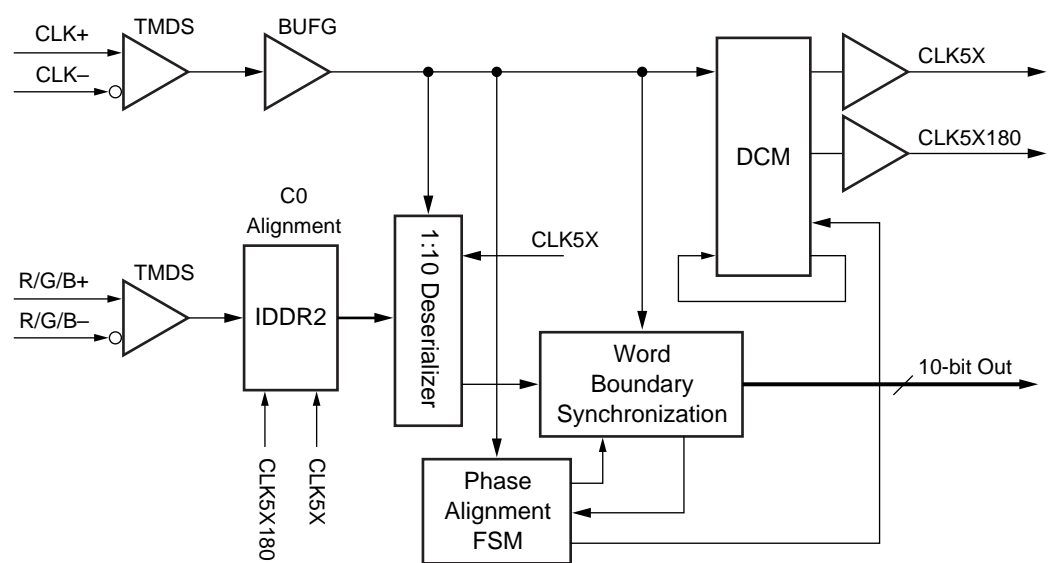
A finite state machine (FSM) can be built to implement the algorithm as illustrated in [Figure 14](#).



X460\_14\_062008

Figure 14: Phase Alignment State Machine

With the phase alignment problem resolved, the complete CDR circuit is now ready for construction as shown in [Figure 15](#).



X460\_15\_070208

Figure 15: Spartan-3A FPGA CDR

The operation of the CDR logic is as follows:

1. The pixel clock and data are fed into the FPGA through a pair of dedicated internal TMDS input buffers.
2. The clock is fed into the BUFG and then into the DCM.
3. The pixel clock is buffered by the BUFG and drives the FPGA logic.
4. The data is fed into an IDDR2 register first before it is passed to a 1:10 deserializer. The DDR technique reduces the bit sampling rate by half.
5. The DCM CLKFX and CLKFX180 ports output CLK5X and CLK5X180, and both provide 5X multiplication. Together, they provide the actual 10X bit rate.
6. The IDDR2 register has a unique C0 alignment feature that allows synchronization of the CLKFX180 sampled data in the CLKFX domain. Thus, the deserializer logic only needs to deal with the CLKFX domain.
7. The rest of the FPGA logic is divided into the CLK5X and the pixel clock domains.
8. CLK5X is needed only to drive the 1:10 deserializer while everything else is driven by the pixel clock. Using CLK5X to drive the 1:10 deserializer greatly increases the timing slack and makes the implementation achievable in the FPGA.
9. A set of distributed RAMs is used in the deserializer as a dual port FIFO to cross the data from CLKFX domain back to the pixel domain.
10. The word boundary detection is also based on the detection of control tokens. It works together with the phase alignment unit to provide the final recovered 10-bit data.

This CDR circuit is duplicated for each of the three data channels. Each CDR unit has its own TMDS input buffer, IDDR2 register, 10:1 deserializer, DCM, word boundary detector, and phase alignment state machine. By using the DDR technique, the 10:1 deserialization is clocked by a pair of 5X clocks. There is no need for 10X clocking using this approach.

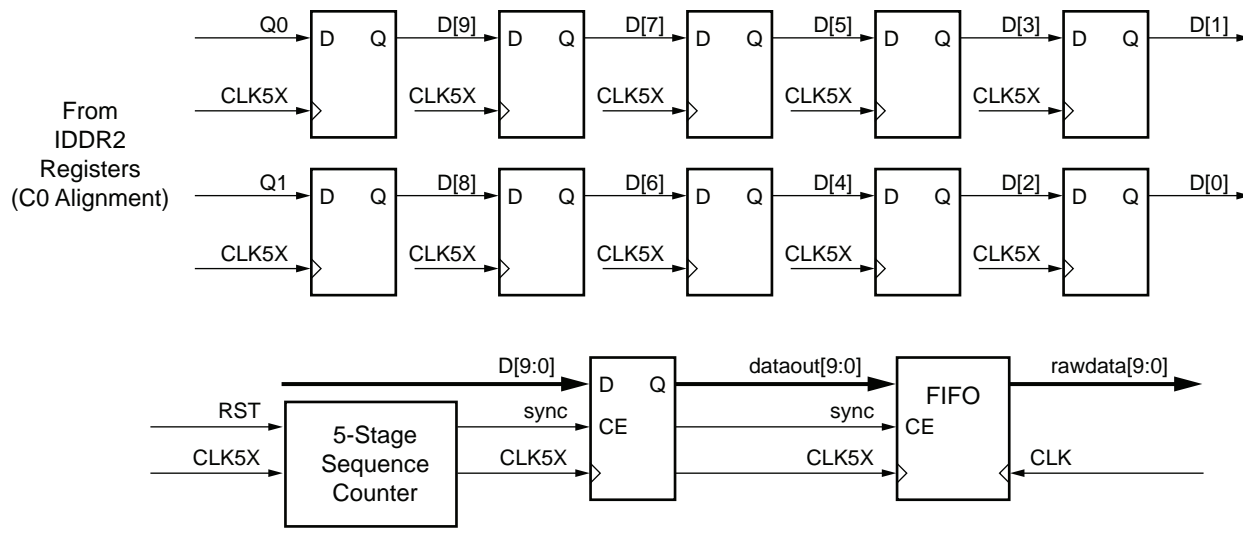
The *Spartan-3A FPGA Family: Data Sheet* [\[Ref 6\]](#) gives the typical phase shift range of the DCM in number of steps:  $\pm [\text{INTEGER} (15 \times (\text{TCLKIN} - 3 \text{ ns}))]$ . TCLKIN is the pixel clock period, which is about 13 ns. Thus, the DCM can shift roughly  $\pm 150$  or a total of 300 steps. Each step is averaged to between 15 ps to 35 ps, So the actual phase shift range is from  $15 \times 300 = 4500 \text{ ps}$  (4.5 ns) to  $35 \times 300 = 10500 \text{ ps}$  (10.5ns). Ensuring the two UIs in the HD video ( $1.3 + 1.3 = 2.6\text{ns}$ ) are adequately covered.

After all three channels are aligned, a group of three 10-bit data is sent to the last two processing stages: the channel deskew and the 10B/8B decoder to generate the final 24-bit color components and the video timing information.

### 1:10 Deserializer

As a part of the CDR circuit, the 1:10 deserializer illustrated in [Figure 16](#) is built using a series of pipeline registers.





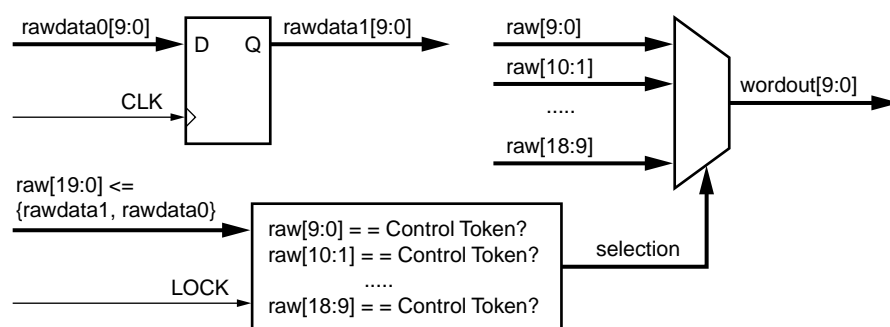
X460\_16\_062408

Figure 16: 1:10 Deserializer

Inputs Q0 and Q1 directly link to IDDR2 outputs Q0 and Q1. By using the C0 alignment feature, Q0 and Q1 are synchronized to CLK5X. A 5-stage sequence counter generates a sync pulse used to latch all pipeline outputs into a 10-bit output. An asynchronous FIFO is used to transfer the data from the CLK5X into the CLK domain. The FIFO is constructed using distributed RAMs described in [Serializer and Clocking Schemes](#). The FIFO output forms a stream of 10-bit raw characters ready for word boundary detection.

### Word Boundary Detection

The word boundary detection is another key part of the CDR circuit. The deserializer is not aware of word boundaries, so the raw characters provided by the deserializer must be reshuffled to regenerate word-aligned symbols. This is done by synchronizing the output words to within two raw characters of the control tokens. [Figure 17](#) shows how this process is implemented.



X460\_17\_070208

Figure 17: Word Boundary Detection

After the 10-bit word is regrouped, it is output to the phase alignment unit. Once the phase alignment is achieved (that is, a valid eye open is detected), a LOCK signal is generated to lock the word boundary. The 10-bit word is then considered valid for the second receiver stage for channel to channel deskew. The CDR processing flow is completed.

## DCM Reset

The DCM plays a vital role in the CDR circuit. However, the clock coming from a DVI or HDMI cable is not always steady enough to satisfy CLKIN jitter and period requirements of the DCM. Consequently, the DCM might go out of lock or invalidate its CLK5X outputs. This can occur in two common situations:

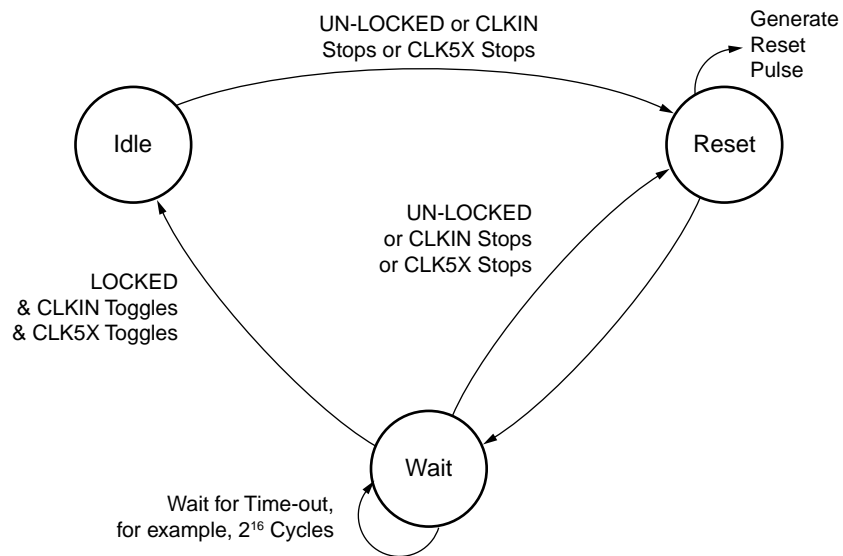
- The cable is unplugged so that there is no more valid input on CLKIN
- The CLKIN frequency changes due to a screen mode switch, for example, 480p is switched to 1080i

When either situation occurs, a reset is desirable. The *Spartan-3A FPGA User Guide* [Ref 7] describes two sets of status indicators that are available from the DCM:

- LOCKED output: Indicates if the DCM as a whole is locked into CLKIN
- STATUS port: Indicates the status of individual DCM components
  - STATUS[1]—Whether CLKIN is toggling or not
  - STATUS[2]—Whether the DFS outputs (CLK5X and CLK5X180) toggle or not

These indicators have to be monitored together to precisely generate the desired reset at the right moment. The reset pulse must last at least three cycles of CLKIN and is only effective if CLKIN actually recovers and retoggles. A simple state machine with three states (Figure 18) is built to actively monitor these cases and generate the proper reset pulse.

Because the TMDS clock is not always stable, a separate steady clock source is needed to drive the state machine.

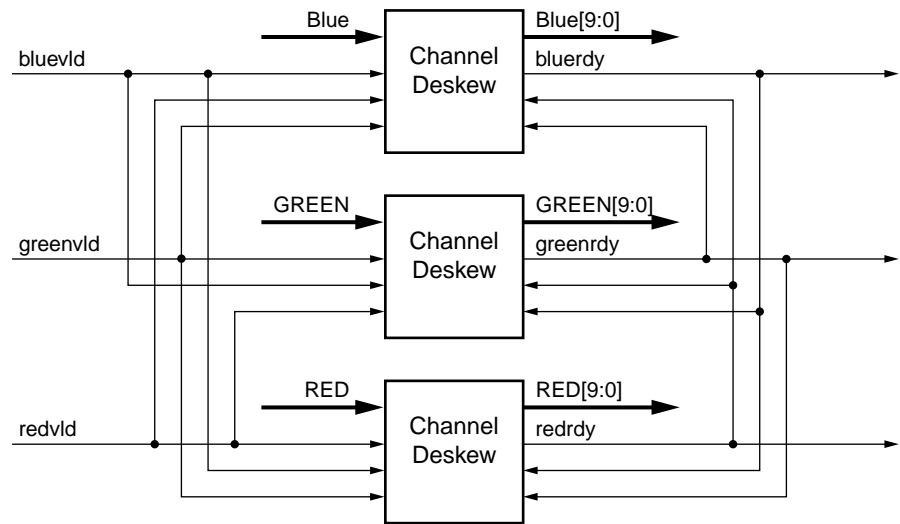


X460\_18\_070208

Figure 18: DCM Reset State

## Channel Deskew

Both DVI 1.0 and HDMI 1.3 specifications require the receiver to tolerate channel-to-channel skew up to  $0.6 T_{\text{pixel}}$ . The channel deskew circuit shown in Figure 19 uses a handshake negotiation scheme between all three channels to remove any skew beyond the specification limit.



X460\_19\_071008

Figure 19: Channel-to-Channel Deskew

Each channel receives a signal from the phase alignment units to indicate if the incoming 10-bit symbols are valid. If all channels are valid (have achieved phase alignment), a FIFO inside each deskew module starts passing data through (continuously writing in and reading out). When a control token is detected in any of the FIFO outputs, the read out flow is suspended and a ready signal is generated to indicate the arrival of a particular marker in the video stream. The read out flow resumes only when this marker has arrived on all three channels. Thus, the relevant skew has been removed.

The FIFO uses the distributed RAM resources and provides a depth of 16 words. Theoretically, the circuit can remove up to 16 pixels of channel to channel skew.

### DVI Decoder

The DVI decoding process is simple, having only two steps:

1. Detect the control tokens
  - a. VDE is driven Low indicating a blanking period
  - b. The two-bit control bus (HSYNC and VSYNC) is remapped for the blue channel
2. Decode
  - a. VDE is driven High
  - b. 10-bit TMDS symbols are converted back to 8-bit pixel components

The decode algorithm is defined in the DVI specification.

### HDMI Decoder

The HDMI decoding process is more complex. Besides video pixels and control tokens, auxiliary data in the data island and guardband tokens must be processed. The processing flow for decoding HDMI is:

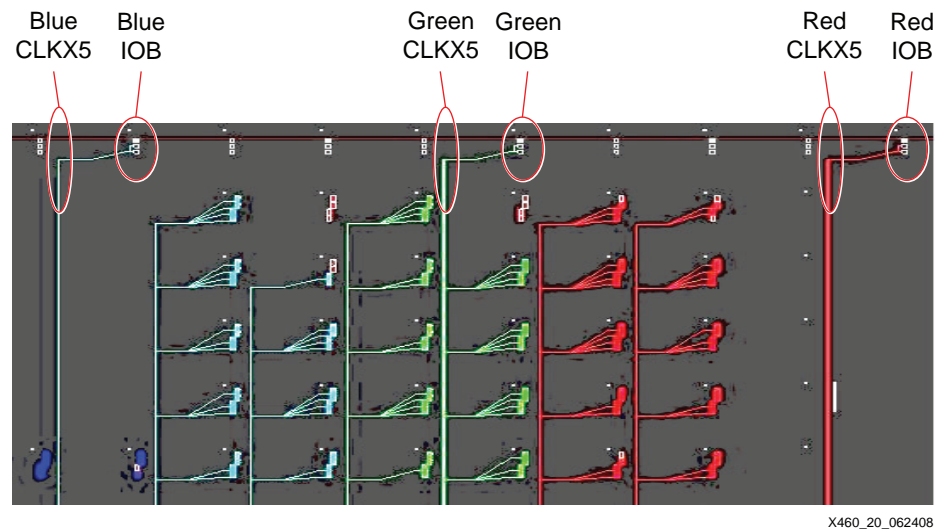
1. Detect the control tokens
  - a. VDE is driven Low indicating a blanking period
  - b. ADE is also driven Low
  - c. The two-bit control bus is remapped:
    - HSYNC and VSYNC for the blue channel
    - CTL0 and CTL1 for the green channel

- CTL2 and CTL3 for the red channel
  - Video preamble indicating the beginning of the video period
  - Data island preamble flag indicating the beginning of the data island period
2. Decode
- a. If the video preamble flag is set:
    - VDE is driven High indicating an active video period
    - ADE is driven Low
    - 10-bit TMDS symbols are converted back to 8-bit pixel components
  - b. If the audio preamble flag is set:
    - ADE is driven High indicating a data island period
    - VDE is driven Low
    - 10-bit TERC4 symbols are converted back to 4-bit auxiliary data

This decode algorithm is defined in the HDMI specification.

### Floorplanning

Like the transmitter design, the three deserializers in the receiver require careful placement because of their fast operating speed at CLK5X. It is also important to group each deserializers into a small rectangle near its associated IOB. [Figure 20](#) shows an example of this kind of placement.



**Figure 20: Deserializer Placements**

The corresponding UCF statements for the deserializer placements are:

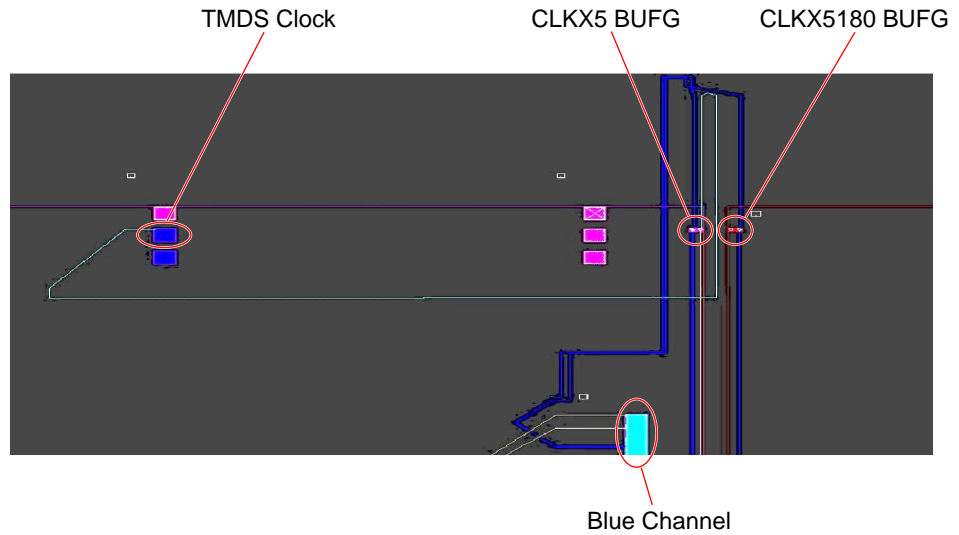
```

INST "hdmi_rx0/dec_b/des/*" AREA_GROUP = "AG_des_b" ;
INST "hdmi_rx0/dec_g/des/*" AREA_GROUP = "AG_des_g" ;
INST "hdmi_rx0/dec_r/des/*" AREA_GROUP = "AG_des_r" ;
AREA_GROUP "AG_des_b" RANGE = SLICE_X48Y134:SLICE_X51Y143 ;
AREA_GROUP "AG_des_g" RANGE = SLICE_X52Y134:SLICE_X55Y143 ;
AREA_GROUP "AG_des_r" RANGE = SLICE_X56Y134:SLICE_X59Y143 ;

```

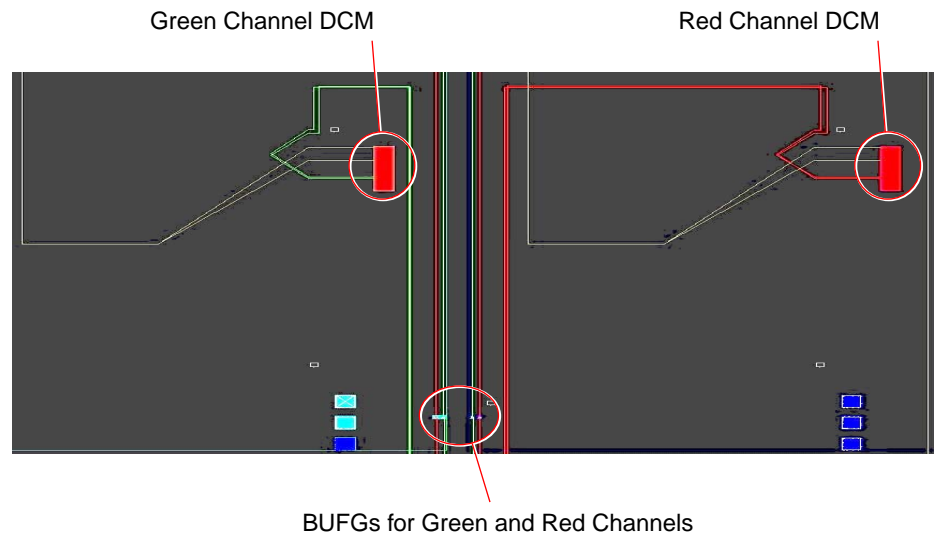
Each deserializer is only 4x12 slices.

Like the transmitter, the two BUFs for each channel's CLK5X and CLK5X180 need to be placed on the same side of the FPGA. Because each channel has its own DCM, the BUFs should also be on the same side as their source DCM. [Figure 21](#) and [Figure 22](#) show an example of this placement for all three channels.



X460\_21\_070208

Figure 21: DCM/BUFG Placements for the Blue Channel



X460\_22\_070708

Figure 22: DCM/BUFG Placements for the Red and Green Channels

The Corresponding UCF statements for the DCM and BUFG placements are:

```

INST "hdmi_rx0/dec_b/dcm_rxclk"      LOC = "DCM_X1Y3";
INST "hdmi_rx0/dec_r/dcm_rxclk"      LOC = "DCM_X2Y0";
INST "hdmi_rx0/dec_g/dcm_rxclk"      LOC = "DCM_X1Y0";

INST "hdmi_rx0/dec_b/clcx5not_bufg"  LOC = "BUFGMUX_X1Y10";
INST "hdmi_rx0/dec_b/clcx5_bufg"     LOC = "BUFGMUX_X2Y11";
INST "hdmi_rx0/dec_r/clcx5not_bufg"  LOC = "BUFGMUX_X1Y0";
INST "hdmi_rx0/dec_r/clcx5_bufg"     LOC = "BUFGMUX_X2Y1";
INST "hdmi_rx0/dec_g/clcx5not_bufg"  LOC = "BUFGMUX_X1Y1";
INST "hdmi_rx0/dec_g/clcx5_bufg"     LOC = "BUFGMUX_X2Y0";
    
```

## Receiver Timing Analysis

Seven clock domains exist in the receiver design. They include the TMDS clock (pixel clock), and CLK5X and CLK5X180 for each of the three channels. However, only the three deserializers contain high-speed paths, and everything else runs at the pixel clock rate. Like the transmitter, the pixel clock is constrained as shown in this UCF example:

```
NET "clk" TNM_NET = "TNM_NET_PCLK";
TIMESPEC "TS_PCLK" = PERIOD "TNM_NET_PCLK" 74.25 MHz HIGH 50% PRIORITY 0;
```

The Xilinx ISE timing analysis tool can trace the pixel clock through relevant DCMs and then automatically adds constraints for the CLK5X clocks. Thus, there is no need to explicitly list the frequency limits of the CLK5X clocks in the UCF.

Like the transmitter design, the receiver uses asynchronous FIFOs to cross data from the CLKX5 domain to the PCLK domain, creating multicycle paths. But the receiver uses three FIFOs, and each of them needs to be separately constrained. For example:

```
#####
# RX multicycle paths
#####
TIMEGRP "rxdes_fifo_b" = RAMS(hdmi_rx0/dec_b/des/dpfo_dout<*>);
TIMEGRP "rxdes_fifo_piped_b" = FFS (hdmi_rx0/dec_b/des/dataout<*>);
TIMEGRP "rxdes_fifo_g" = RAMS(hdmi_rx0/dec_g/des/dpfo_dout<*>);
TIMEGRP "rxdes_fifo_piped_g" = FFS (hdmi_rx0/dec_g/des/dataout<*>);
TIMEGRP "rxdes_fifo_r" = RAMS(hdmi_rx0/dec_r/des/dpfo_dout<*>);
TIMEGRP "rxdes_fifo_piped_r" = FFS (hdmi_rx0/dec_r/des/dataout<*>);

TIMESPEC "TS_rxdes_fifo_b" = FROM "rxdes_fifo_b" TO "rxdes_fifo_piped_b"
TS_PCLK;
TIMESPEC "TS_rxdes_fifo_g" = FROM "rxdes_fifo_g" TO "rxdes_fifo_piped_g"
TS_PCLK;
TIMESPEC "TS_rxdes_fifo_r" = FROM "rxdes_fifo_r" TO "rxdes_fifo_piped_r"
TS_PCLK;
```

## Implementation

This section summarizes the implementation of the [Transmitter Design](#) and the [Receiver Design](#) into real hardware.

### Proof of Concept Boards

Hardware evaluation of the transmitter and receiver designs took place over a span of several months using progressively refined schemes. The initial feasibility test was performed with a Spartan-3A FPGA Starter Kit. A standard DVI cable was cut in half, and the conductors were soldered to a 34-pin IDC connector, which mated with the differential I/O expansion connectors on the Starter Kit. The pixel clock used for this test was derived from the onboard oscillator and did not exactly match the screen mode specifications for the pixel clock. However, this proof of concept worked at a low bit rate and justified the further development of the transmitter and receiver designs as well as more refined hardware solutions.

To enable development and evaluation by multiple users, a small and inexpensive adapter board was fabricated for the Spartan-3A FPGA Starter Kit to offer mechanical stability for the proof of concept assembly. This adapter board, shown in [Figure 23](#), uses an HDMI connector and attaches to either of the differential I/O expansion connectors on the Spartan-3A FPGA Starter Kit.

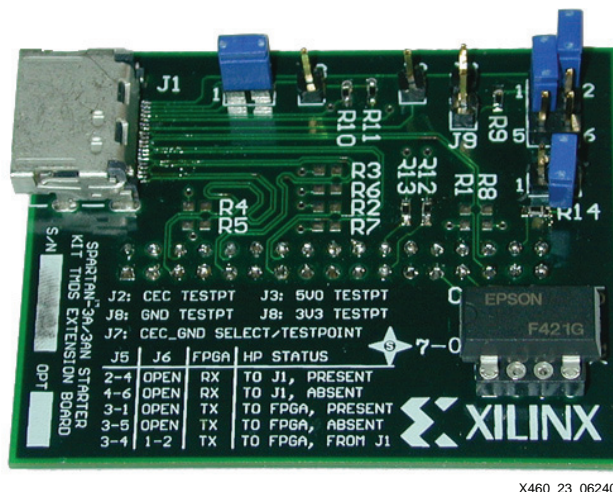


Figure 23: Spartan-3A FPGA Starter Kit Adapter Board

Depending on which assembly options were installed, the adapter board enabled either transmitter or receiver operation. However, the design did not address many DVI and HDMI requirements. Also, the termination resistor placement forced by the intermediate expansion connector—and the frequency response of the expansion connector itself—were impediments to operation at high bit rates.

Signal integrity concerns aside, the standard Spartan-3A FPGA Starter Kit uses a lower speed grade (-4) device, which supports TMDS signaling up to 640 Mb/s (Table 1). The higher speed grade (-5) provides for TMDS signaling up to 700 Mb/s. Both are adequate for SD (480p) video and other low resolution graphics applications such as VGA and SVGA. The higher speed grade (-5) device further enables XGA graphics applications.

The adapter board serves as a temporary solution while the evaluation board is designed for operation across the full range of bit rates.

## Evaluation Board

The goal of the evaluation board is to enable evaluation of TMDS transmitter and receiver functions across a wide range of bit rates. This includes evaluating the internal serializer and deserializer performance, TMDS electrical and protocol compatibility, and auxiliary functions such as data display channel (DDC), hot plug detect (HPD), and consumer electronics control (CEC), which are sideband communication schemes used in DVI and HDMI.

### Evaluation Board Considerations

The evaluation board has with a dedicated transmit port and a dedicated receive port, both using HDMI connectors. It also includes an expansion port to provide single-ended and differential signaling capability at 3.3V.

The HDMI specification was used as the guideline for determining key considerations for the evaluation board. The main considerations are discussed in the following sections.

### TMDS Electrical Considerations

Spartan-3A FPGAs can directly transmit and receive TMDS signals through use of the TMDS\_33 SelectIO mode. The electrical characteristics of this mode are presented in the *Spartan-3A FPGA Family: Datasheet* [Ref 6]. The implementation requirements for the TMDS\_33 SelectIO™ mode are presented in the *Spartan-3 Generation FPGA User Guide* [Ref 7]. Key requirements are:

- TMDS\_33 outputs are supported only on banks 0 and 2 (top and bottom).
- TMDS\_33 outputs require  $V_{CCO} = 3.3V$  and  $V_{CCAUX} = 2.5V/3.3V$ .
- TMDS\_33 inputs require  $V_{CCO} = 2.5V/3.3V$  and  $V_{CCAUX} = 3.3V$ .
- TMDS\_33 signals require  $50\Omega$  termination to  $3.3V$  at the receiver.

These requirements influence power supply design for the system and also place some restrictions on pin assignments. For a simple design that both transmits and receives TMDS signals (such as an evaluation board), it is easiest to use  $3.3V$  for  $V_{CCAUX}$  and for the  $V_{CCO}$  on all four banks of the device. However, the TMDS\_33 requirements allow additional flexibility for  $V_{CCAUX}$  and  $V_{CCO}$  in mixed voltage environments, especially when transmit and receive capabilities are not implemented concurrently.

For SD video support, the lowest speed grade (-4) devices are more than adequate.

Several additional TMDS electrical considerations are addressed at the board level. One of these is the  $100\Omega$  channel differential impedance. This differential impedance includes the differential pairs on a board (at both transmitter and receiver), the connectors, and the cabling.

The HDMI specification defines impedance requirements at several points on the signal path (all  $100\Omega$ , but different tolerances) including the cable and connectors. The tightest tolerance noted in the specification is  $\pm 10\%$ . This tolerance was used when determining the differential pair trace geometry for the evaluation board.

Estimating a differential impedance of  $100\Omega$  is usually not as simple as assuming a differential pair where each conductor has a characteristic impedance of  $50\Omega$ . Under certain circumstances this estimate might be acceptable, but in most situations use an impedance calculator or field solver to determine impedance.

For the evaluation board, where the differential pairs were routed on outer board layers, an approximation for the differential impedance of a microstrip pair was used to estimate the differential impedance (see Equation 1):

$$Z_{diff} = 2Z_0 \left( 1 - 0.48 e^{\left( -0.96 \left( \frac{S}{H} \right) \right)} \right) \quad \text{Equation 1}$$

Where:

$S$  = edge to edge trace/trace separation

$H$  = edge to edge trace/plane separation

A proven board stack-up and trace geometry was used. As a result, the calculated differential impedance for the differential pairs on the evaluation board is approximately  $93\Omega$ , which is within the target of  $100\Omega \pm 10\%$ . Impedance discontinuities on differential pairs routed to the HDMI connectors (and half of the pairs routed to the expansion port) were minimized by routing entirely on the top layer of the board so that no vias are required.

Another board-level consideration is skew minimization. The HDMI specification does not define specific skew requirements. Instead, the specification requires limits on skew as measured at the output connector of a transmitter, and requires tolerance for skew as measured at the input connector of a receiver. Skew might result from delay mismatches in connectors, board routing, package substrates, and actual transmitter/receiver implementation.

At the transmit connector, the HDMI specification requires the intrapair skew to be 15% or less of a bit time (approximately 215 ps at 700 Mb/s) and interpair skew to be 20% or less of a character time (about 2.90 ns at 700 Mb/s). At the receive connector, the HDMI specification requires the intrapair skew tolerance to accommodate the maximum transmit intrapair skew plus 112 ps, and the interpair skew tolerance to accommodate the maximum transmit interpair skew plus 1.78 ns.

In a typical application, the distance between the TMDS device and the connector is quite short, and the number of pairs is low, making it very easy to equalize traces. For the evaluation



board, the single layer routing and consistent trace geometry made it easy to match all TMDS signals to lengths of within 5 mils of each other. As a result, the trace skew becomes a negligible part of the overall skew budget providing as much slack as possible for potential skew sources in the transmitter and receiver.

The length of the pairs routed to the expansion port entirely on the top layer of the board were matched to within 10 mils of each other. The length of the remaining pairs were also matched to 10 mils, but the length of the two sets are not matched to each other.

### Reference Clock Considerations

The HDMI specification discusses data and clock jitter, but does not directly specify any particular requirement for the system clock. Many of the jitter requirements are specified at the connectors, where the jitter is a function of the system clock, as viewed through the FPGA clock network (possibly including the DCM) and the FPGA resources.

The evaluation board accommodates experimentation with two oscillator styles, as shown in Figure 24. The board uses commodity LVCMOS output oscillators but can also support very low jitter differential LVDS output oscillators.

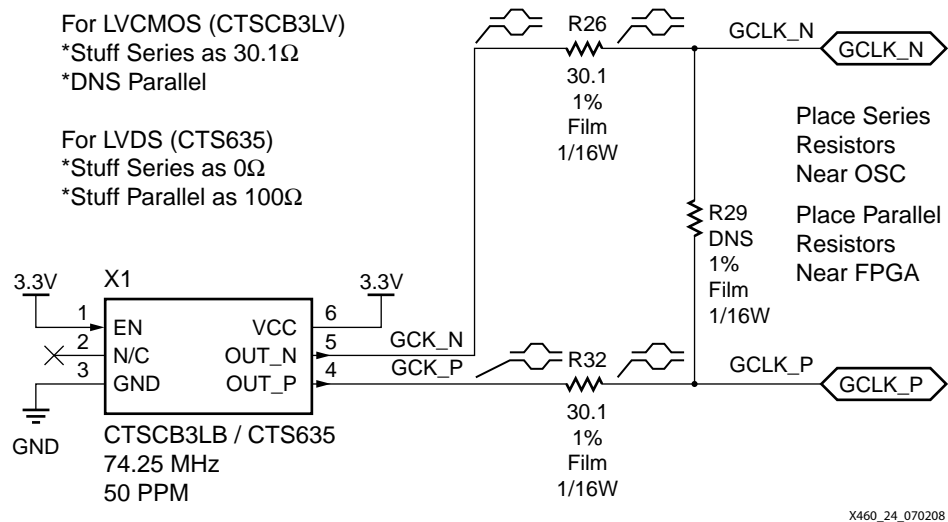


Figure 24: Reference Clock Options on the Evaluation Board

### DDC, HPD, and CEC Considerations

The evaluation board implements DDC compatibility, but compliance to the DDC specification was not a design goal and has not been verified. The evaluation board meets required output and detection levels for the HPD signal. CEC is optional and is not implemented.

### ESD Protection Considerations

For many HDMI applications, ESD protection is a critical concern. ESD requirements are set in the HDMI Compliance Test Specification, which calls implementations to meet the requirements set in *IEC 61000-4-2 Level 4* [Ref 8]. These requirements correspond to  $\pm 8$  KV contact discharge and  $\pm 15$  KV air discharge.

The Spartan-3A device ESD rating is 2 KV using the human body model. This level of protection is intended to protect the device from discharge that might occur in a typical manufacturing environment, and is also adequate for applications enclosed within a protected chassis. For most consumer applications, this level of protection is not sufficient. To protect the Spartan-3A device, external protection diodes are added to raise the ESD protection level to meet or exceed the IEC 61000-4-2 Level 4 requirements.

## Evaluation Board Preliminary Results

The preliminary results from the evaluation board proved robust operation of TMD5 transmit and receive functions across all tested bit rates. The measured eye diagrams support operation at SD, VGA, SVGA, and XGA bit rates.

Figure 25 shows the evaluation board.

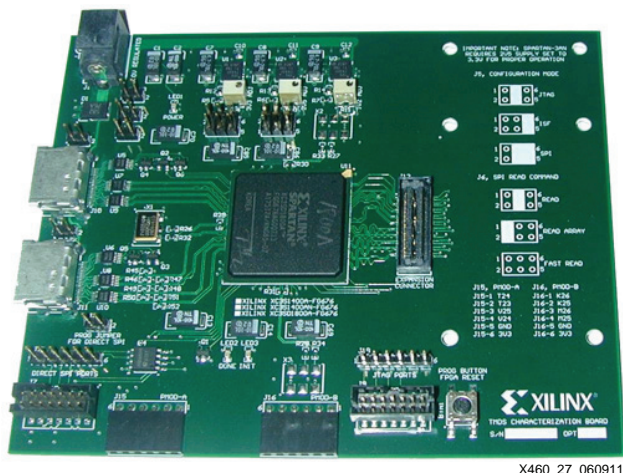


Figure 25: TMD5 Evaluation Board

## Reference Design

Using the information provided in the [Evaluation Board](#) section, a reference application design was created for demonstration and evaluation purposes.

### Reference Design Goals

The goal of the reference design is to provide a simple but highly effective and visible platform to validate the soft DVI/HDMI transceiver.

### AV Pass-Through with Overlay

Based the idea of what you see is what you get, an audio-video pass-through design simultaneously exercises the transmitter and the receiver. [Figure 26](#) illustrates the design concept. To prove the video stream is fully decoded in the receiver and re-encoded in the transmitter, an image of the Spartan-3 generation logo is superimposed on the decoded pixel data of a digital video recording as it passes between the transmitter and receiver. The resulting composite movie is displayed on a high-definition monitor to confirm the Spartan-3 FPGA has the throughput to support the high data rates required for processing HDMI or DVI video streams.

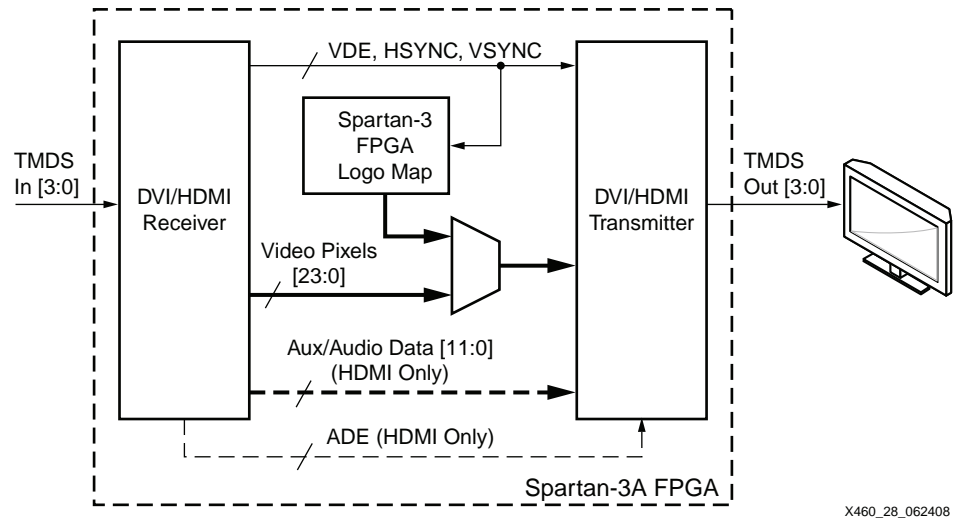


Figure 26: AV Pass-Through Evaluation with Graphics Overlay

### Evaluation Design Results

The Spartan-3 logo is overlaid on the fly with the received video pixels and played back onto a standard HD LCD display panel.

The design demonstrates the great value of the DVI/HDMI FPGA solution: Reception, Customized Manipulation and Transmission in a single chip.

### Resource Utilization

The FPGA resources used by the reference designs described in this document are listed in [Table 3](#).

Table 3: FPGA Resources Used by Each Design

	DVI Transmitter (alone)	HDMI Transmitter (alone)	DVI Receiver (alone)	HDMI Receiver (alone)
<b>LUTs</b>	357	507	995	1183
<b>Flip-Flops</b>	167	184	555	579
<b>BUFGs</b>	3	3	7	7
<b>DCMs</b>	1	1	3	3
<b>Block RAM</b>	0	0	0	0

The final resource usage is very application dependent. For instance, in the pass-through evaluation design, the transmitter and the receiver share the same clocking resources for the pixel clock and X5 clocks. So the transmitter's actual BUFG and DCM count is the same as the receiver alone.

## Conclusion

The Spartan-3A FPGA design and evaluation board described in this application note demonstrates a TMDS video connectivity solution useful for consumer DVI/HDMI applications.

## Design Limitations and Disclaimers

This application note provides only the physical layer solution for the DVI/HDMI interface. It does not address all required or optional protocol aspects for achieving compliance:

Required DDC sideband protocol (I<sup>2</sup>C)

- Monitor discovery for the transmitter
- EDID data structure for the receiver

Optional HDCP content protection

- Leveraging the DDC side band protocol to exchange encryption keys
- Content encryption engine for the transmitter
- Content de-encryption engine for the receiver

Required HDMI audio/auxiliary packaging and channelization

- See Chapters 7 and 8 in the HDMI specification, version 1.3

Optional deep color mode

- Beyond 24-bit color
- HDMI specification, version 1.3 and above only

Optional Consumer Electronics Control (CEC)

- HDMI specification, version 1.3, supplement 1

There are, however, no known technical difficulties for FPGA users to implement these features themselves.

## Limited Characterization

These test results were derived from a transmitter only Spartan-3A FPGA design using an onboard oscillator as the pixel clock source. The video source signal was provided by a SMPTE HD color bar generator. The test instrument used was a Lecroy SDA9000 scope with a DVI/HDMI characterization software package installed.

The characterization tests are based on DVI 1.0 and HDMI 1.3 specifications. The receiver was not characterized due to the limitations of the available test equipment.

### Transmitter Voltage Test

Voltage levels must fall within the acceptable range for  $V_H$  and  $V_L$  defined here:

- $V_H = AV_{CC} \pm 10 \text{ mV}$
- $(AV_{CC} - 600 \text{ mV}) \leq V_L \leq (AV_{CC} - 400 \text{ mV})$ ; where  $AV_{CC} = 3.3\text{V}$

Result: PASS

Figure 27 illustrates the measurement for this test. The base and top histogram peaks reflect the  $V_L$  and  $V_H$  ranges.

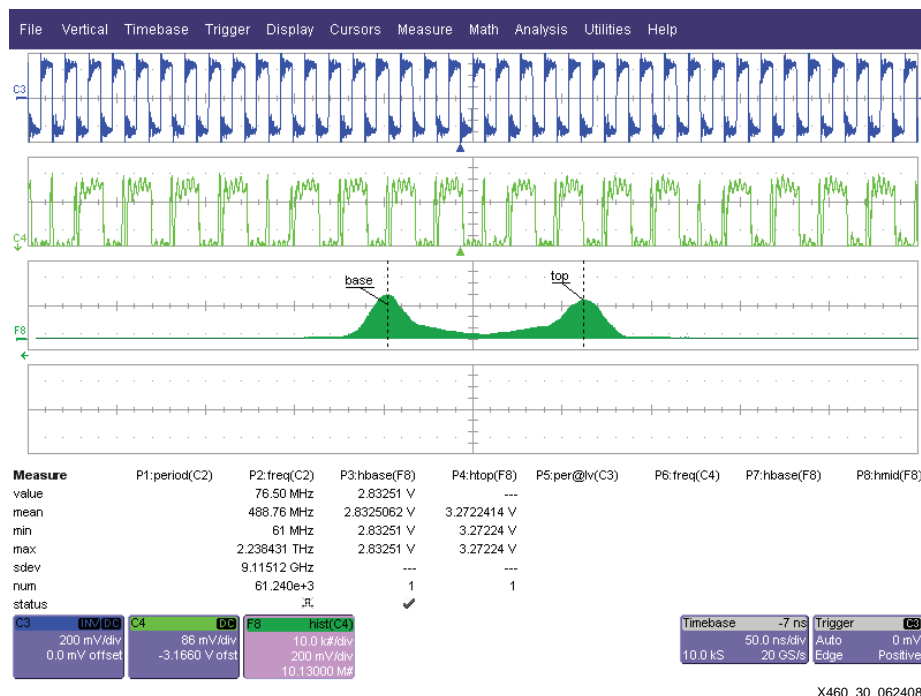


Figure 27: Single-Ended  $V_L$  (Base) and  $V_H$  (Top)

### Data Channel Rise and Fall Time Test

The data waveform must meet the rise and fall times specified here:

- $75 \text{ ps} \leq \text{Rise time, } T_{\text{BIT}} = 1.3 \text{ ns}$
- $\text{Fall time} \leq 0.4T_{\text{BIT}}, T_{\text{BIT}} = 1.3 \text{ ns}$

Result: PASS

Figure 28 illustrates the measurement for this test.

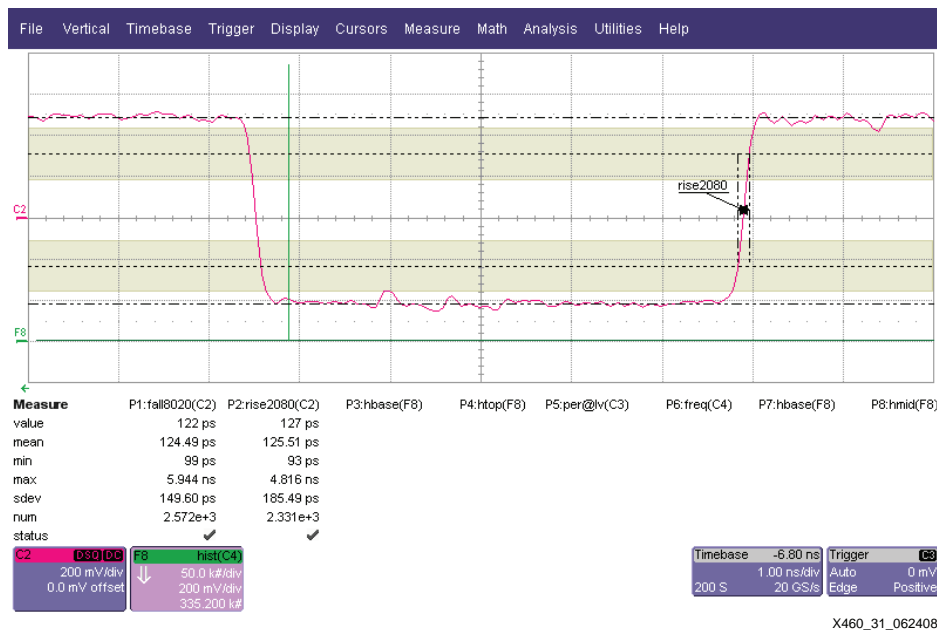


Figure 28: TMD5 Rise and Fall Times On the Blue Channel

## Interpair Skew Test Between Data Channels

The skew between any pair of data channel must fall within the values specified here:

- Skew  $\leq 0.20T_{\text{pixel}}$ ; where  $T_{\text{pixel}} = 13 \text{ ns}$

Result: PASS

Figure 29, Figure 30, and Figure 31 illustrate the measurements for the red, green, and blue channels.

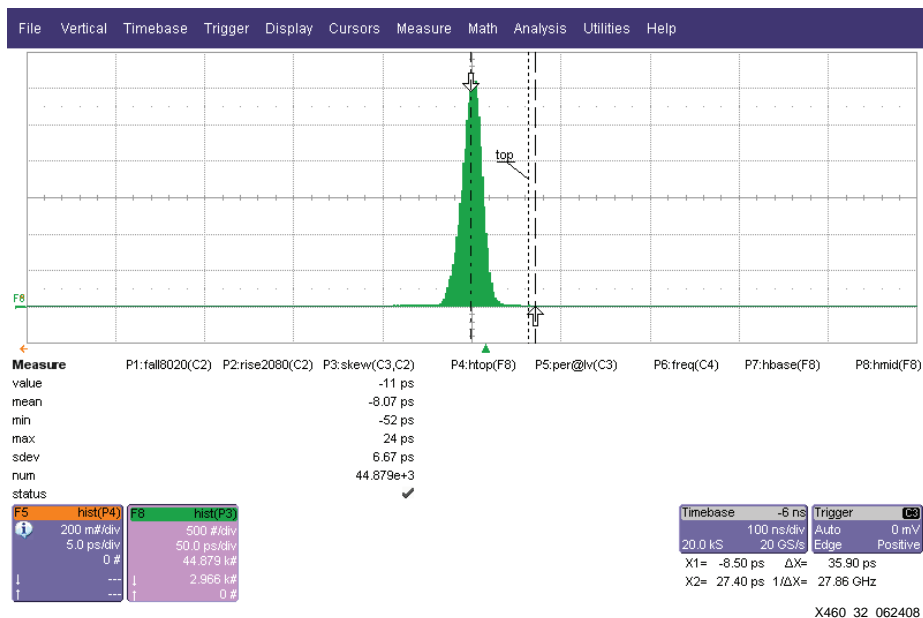


Figure 29: Interpair Skew Between Blue and Green Channels

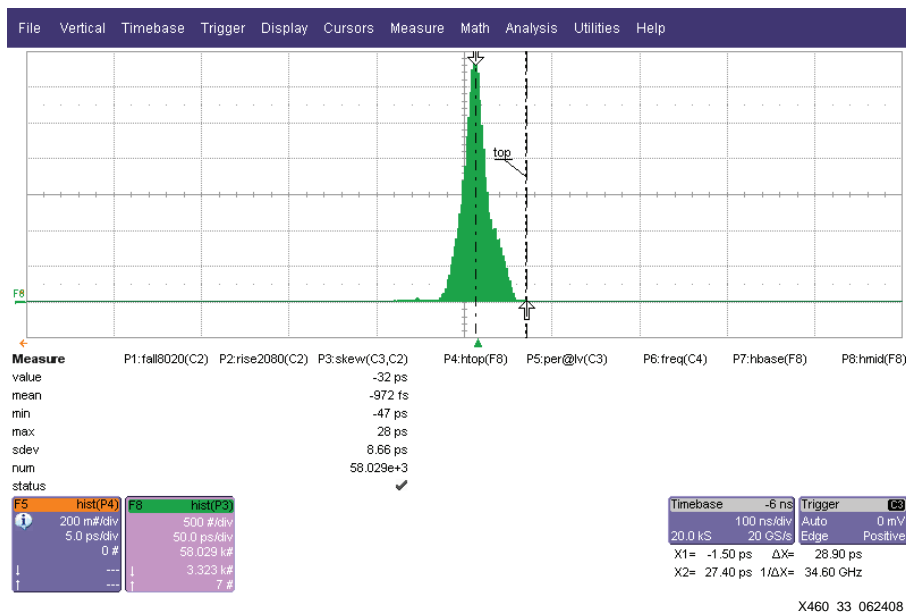


Figure 30: Interpair Skew Between Blue and Red Channels

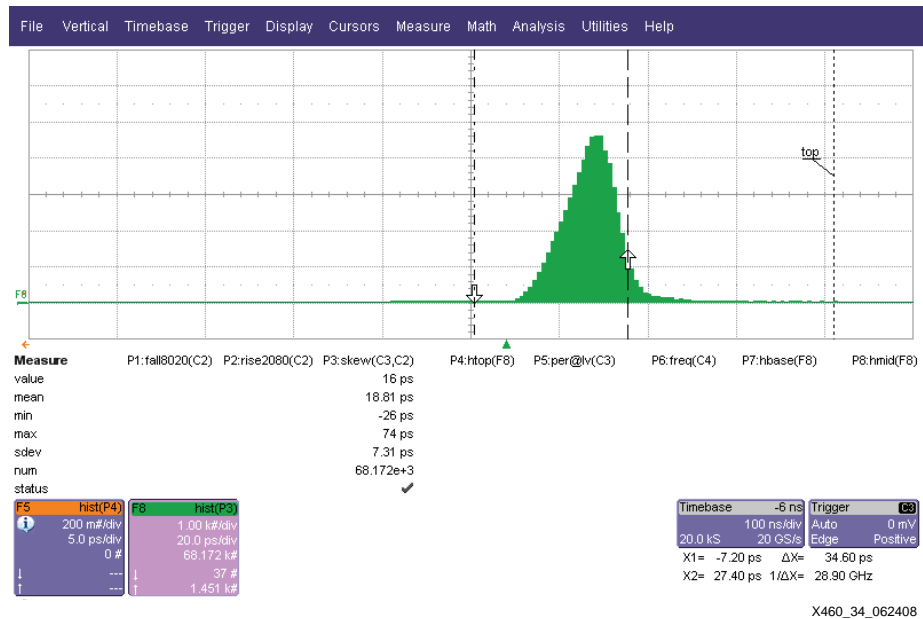


Figure 31: Interpair Skew Between Green and Red Channels

## Clock Duty Cycle Test

The clock duty cycle must fall within the value specified here:

- 40% ≤ Clock Duty Cycle ≤ 60%

Result: PASS

Figure 32 illustrates the measurement for the test.

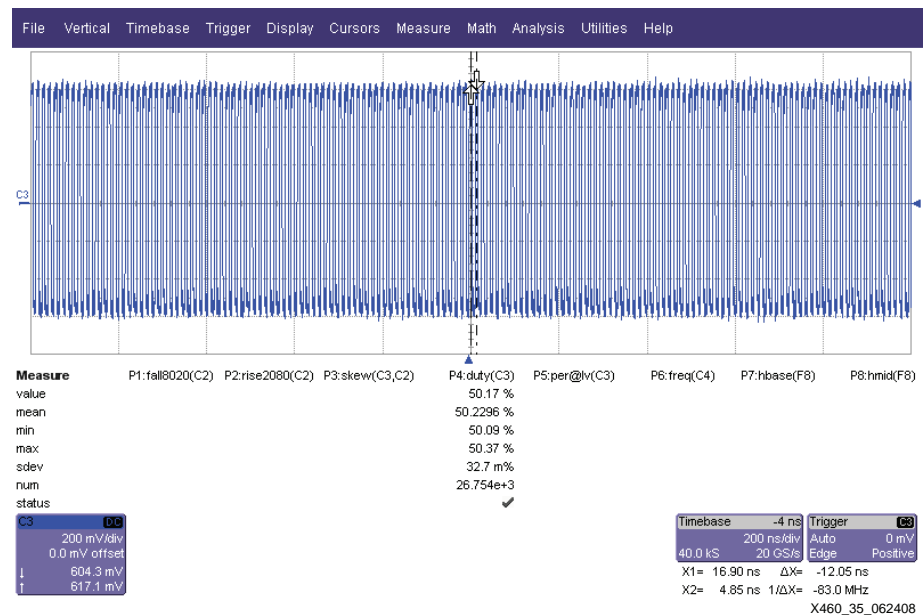


Figure 32: TMD5 Clock Duty Cycle

## Clock Jitter Test

The clock jitter must fall within the value specified here:

- Total Jitter  $\leq 0.25T_{bit}$ ; where  $T_{bit} = 1.3 \text{ ns}$

Result: PASS

The jitter measurement is set to be relative to an Ideal Recovery Clock as defined in both DVI and HDMI specifications [Ref 2], [Ref 3]. Figure 33 illustrates the measurement for this test.

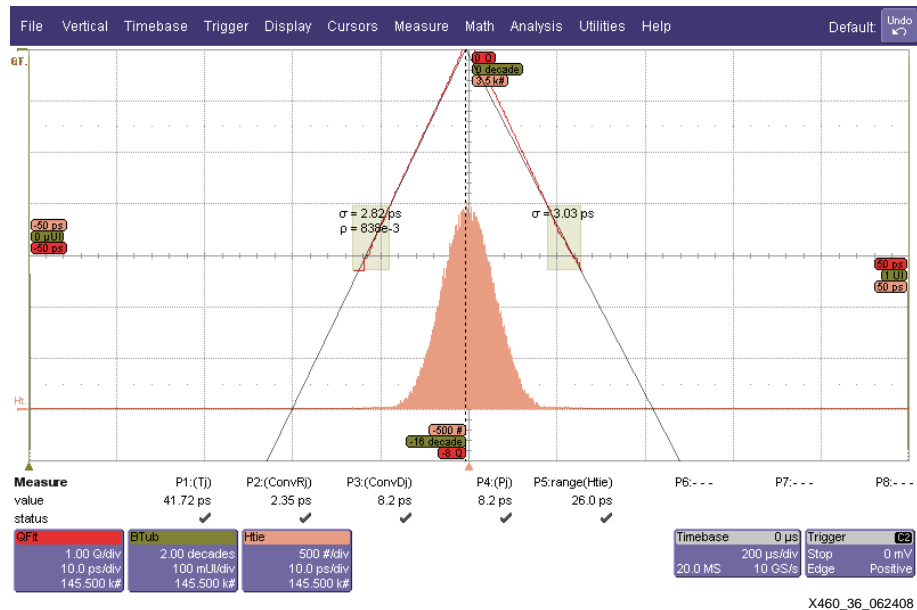


Figure 33: TMDS Clock Jitter

## Data Eye Diagram Test

The waveform must fall outside the eye mask defined in the DVI and HDMI specifications. The pass criteria for this test is:

- No mask violation

Result: PASS

The eye mask is automatically set using the Lecroy DVI/HDMI software package, meeting specifications of both DVI and HDMI. Figure 34, Figure 35, and Figure 36 illustrate the eye diagrams the red, green, and blue channels, respectively.



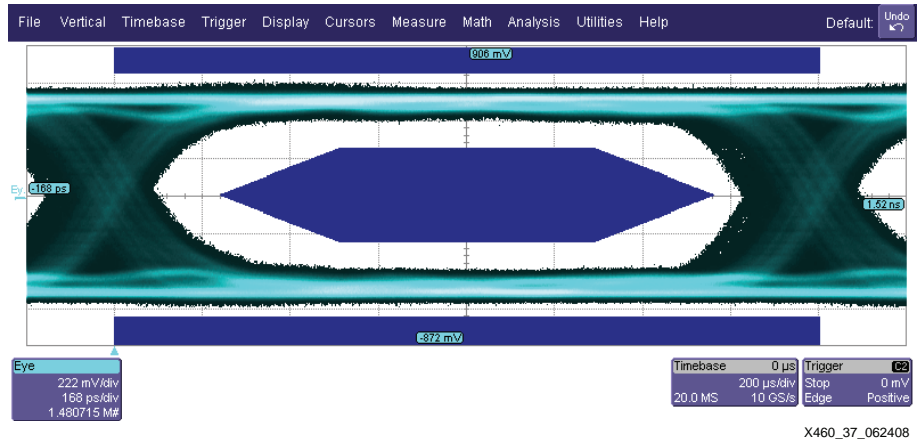


Figure 34: Red Channel Data Eye

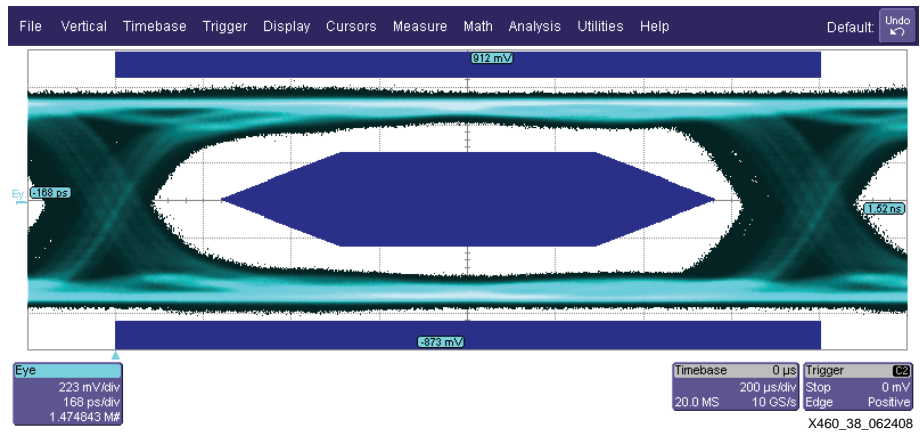


Figure 35: Green Channel Data Eye

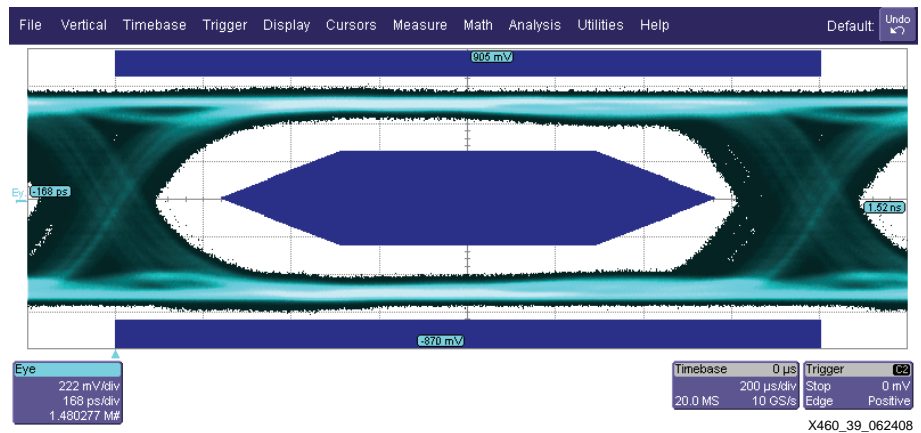


Figure 36: Blue Channel Data Eye

## Reference Design Files

Design files are available in xapp460.zip and can be downloaded from <https://secure.xilinx.com/webreg/clickthrough.do?cid=111774>.

Table 4: Reference Design Files Contained in xapp460.zip

Module Name	Path and Filename
<b>DVI Transmitter</b>	
DVI transmitter top Wrapper	dvi_demo/rtl/tx/dvi_encoder.v
DVI encoder	dvi_demo/rtl/tx/encode.v
30:3 Serializer	dvi_demo/rtl/tx/serdes_4b_10to1_fifo.v
<b>DVI Receiver</b>	
DVI receiver top wrapper	dvi_demo/rtl/tx/dvi_decoder.v
DVI decoder instantiating the CDR and channel deskew circuits	dvi_demo/rtl/tx/decode.v
Channel deskew module	dvi_demo/rtl/tx/chnlbond.v
DCM reset state machine	dvi_demo/rtl/tx/dcminit.v
Phase alignment state machine	dvi_demo/rtl/tx/phsaligner.v
1:10 deserializer	dvi_demo/rtl/tx/tmds_1c_1to10.v
<b>DVI Common Modules</b>	
Width configurable distributed RAM	dvi_demo/rtl/common/DRAM16XN.v
Video Timing Controller	dvi_demo/rtl/common/timing.v
SMPTE HD color bar generator	dvi_demo/rtl/common/hdclrbar.v
<b>DVI Evaluation</b>	
DVI Evaluation design	dvi_demo/rtl/dvi_demo.v
DVI transmitter only instantiating its own video timing controller and HD color bar generator	dvi_demo/rtl/dvitz_demo.v
<b>HDMI Transmitter</b>	
HDMI transmitter top Wrapper	hdmi_demo/rtl/tx/hdmi_encoder.v
DVI encoder	hdmi_demo/rtl/tx/encode.v
30:3 Serializer	hdmi_demo/rtl/tx/serdes_4b_10to1_fifo.v
<b>HDMI Receiver</b>	
HDMI receiver top wrapper	hdmi_demo/rtl/rx/hdmi_decoder.v
HDMI decoder instantiating the CDR and channel deskew circuits	hdmi_demo/rtl/rx/decode.v
Channel deskew module	hdmi_demo/rtl/rx/chnlbond.v
DCM reset state machine	hdmi_demo/rtl/rx/dcminit.v
Phase alignment state machine	hdmi_demo/rtl/rx/phsaligner.v
1:10 deserializer	hdmi_demo/rtl/rx/tmds_1c_1to10.v
<b>HDMI Common Modules</b>	
Width configurable distributed RAM	hdmi_demo/rtl/common/DRAM16XN.v

Table 4: Reference Design Files Contained in xapp460.zip (Cont'd)

Module Name	Path and Filename
Width configurable delay lines using SRL16	hdmi_demo/rtl/common/srldelay.v
HDMI Evaluation design	hdmi_demo/rtl/hdmi_demo.v

The reference design matrix is shown in [Table 5](#).

Table 5: Reference Design Matrix

<b>General</b>	
Developer Name	Xilinx
Target Devices (stepping level, ES, production, speed grades)	Spartan-3A Family FPGAs
Source Code Provided	Yes
Source Code Format	Verilog
Design Uses Code/IP from an Existing Reference Design/Application Note, 3rd Party, or CORE Generator software	No
<b>Simulation</b>	
Functional Simulation Performed	Yes
Timing Simulation Performed	Yes
Testbench Used for Functional Simulations Provided	No
Testbench Format	Verilog
Simulator Software Used/Version (e.g., ISE software, Mentor, Cadence, other)	Cadence IUS version 6.2
SPICE/IBIS Simulations	No
<b>Implementation</b>	
Synthesis Software Tools Used/Version	XST
Implementation Software Tools Used/Versions	ISE 10.1i
Static Timing Analysis Performed	Yes
<b>Hardware Verification</b>	
Hardware Verified	Yes
Hardware Platform Used for Verification	Spartan-3A TMDS Characterization Board

## References

1. CEA-861-D, *A DTV Profile for Uncompressed High Speed Digital Interfaces*, Consumer Electronics Association, <http://www.ce.org>.
2. *Digital Visual Interface, Revision 1.0*, Digital Display Working Group, <http://www.ddwg.org/>.
3. *High-Definition Multimedia Interface Specification Version 1.3a*, HDMI Licensing, LLC, <http://www.hdmi.org>.
4. [XAPP224](#), *Data Recovery*.
5. [XAPP861](#), *Efficient 8X Oversampling Asynchronous Serial Data Recovery Using IDELAY*.
6. [DS529](#), *Spartan-3A FPGA Family: Data Sheet*.
7. [UG331](#), *Spartan-3 Generation FPGA User Guide*.
8. IEC 61000-4-2, *Electromagnetic compatibility (EMC)- Part 4-2: Testing and measurement techniques - Electrostatic discharge immunity test*, Commission Electrotechnique Internationale, <http://www.iec.ch/>.

## Revision History

The following table shows the revision history for this document.

Date	Version	Description of Revisions
07/25/08	1.0	Initial Xilinx release.
06/24/11	1.1	Changed 750 Mb/s to 700 Mb/s in <a href="#">Summary</a> . Deleted first row and notes from <a href="#">Table 1</a> , and revised 666 Mb/s to 700 Mb/s and 622 Mb/s to 640 Mb/s in the remaining two rows. Modified <a href="#">step 4</a> in <a href="#">Clock and Data Recovery</a> . Modified first sentence in <a href="#">Floorplanning</a> . Revised 622 Mb/s to 640 Mb/s and 666 Mb/s to 700 Mb/s, and deleted last sentence from fourth paragraph in <a href="#">Proof of Concept Boards</a> . Deleted first sentence from third paragraph in <a href="#">TMDS Electrical Considerations</a> . Revised 200 ps at 742.5 Mb/s to 215 ps at 700 Mb/s and 2.70 ns at 742.5 Mb/s to 2.90 ns at 700 Mb/s in tenth paragraph under <a href="#">TMDS Electrical Considerations</a> . Deleted last sentence from first paragraph and second paragraph in <a href="#">Evaluation Board Preliminary Results</a> . Deleted Expansion Board Considerations section. Deleted third sentence from first paragraph in <a href="#">Limited Characterization</a> .

## Notice of Disclaimer

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of the Limited Warranties which can be viewed at <http://www.xilinx.com/warranty.htm>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in Critical Applications: <http://www.xilinx.com/warranty.htm#critapps>.