VITESSE®

Equalizing PCI Express Gen 1, 2, and 3 Channels with the VSC3308 and VSC3316

Application Note

VPPD-02558
Revision 1.0
March 2010

Vitesse
Corporate Headquarters
741 Calle Plano
Camarillo, California 93012
United States

www.vitesse.com

Contents

Revision History 3

1 Introduction 4

2 PCI Express Repeater Requirements 4

2.1 Spread-Spectrum Clocking 4
2.2 Beacon 5
2.3 Electrical Idle 5
2.4 Receiver Detect 5
2.5 Dynamic Rate Negotiation 5
2.6 Transmitter De-Emphasis 6
2.7 PCIe Gen 3 Dynamic Equalization Negotiation 6

3 Meeting the Requirements for a PCIe Gen 3 Buffer/Repeater ... 6

3.1 Spread-Spectrum Clocking, Beacon, and Dynamic Rate Negotiation 7
3.2 Electrical Idle 7
3.3 Receiver Detect 7
3.4 Transmitter De-emphasis 8
3.5 PCIe Gen 3 Dynamic Equalization Negotiation 9

4 PCIe Demonstration of the VSC3316 10

5 Conclusions 14

6 APPENDIX: Source Code for Receiver Detect Firmware 15

Figures

Figure 1. PCIe Test Setup with the VSC3316 11
Figure 2. PCIe to SMA Adapter 11
Figure 3. Desktop PC with PCIe to SMA Adapter 12
Figure 4. SMA to 28AWG Cable Adapter 13
Figure 5. 2-Slot PCIe Backplane Connecting to Video Card 13

Revision History

Revision	Date	Description
Rev 1.0	March 5, 2010	First release of this document

1 Introduction

PCI Express (PCIe) is designed for operation over channels that include up to 20 inches of printed circuit board (PCB) transmission lines and two connectors. As the data rates have increased from PCIe Gen 1 at 2.5 gigabits per second (Gbps) to PCIe Gen 2 at 5.0 Gbps the challenges associated with accurate transmission of the data over 20 inches of PCB have been met by adding limited pre-emphasis to PCIe compliant transmitters. At the time of this writing, PCIe Gen 3 at 8.0 Gbps is still under development, but current plans require expanded transmitter pre-emphasis as well as adjustable receiver equalization in order to function with 20 inches of PCB.

Blade servers and other applications have adopted PCIe for transmitting high-speed data over backplanes and cables that exceed the PCIe standard maximum of 20 inches of PCB. For these applications it is generally necessary to use non-PCIe standard techniques to compensate for the increasing signal degradation caused by the longer channels. One such technique is to place devices in the channel that can compensate for added signal distortion through equalization and regeneration of the signals. Devices of this sort are generally referred to as PCIe buffers or PCIe repeaters.

The purpose of this application note is to outline the requirements for a PCIe buffer/repeater and document successful demonstrations of the VSC3316 16 x 16 crosspoint switch as a buffer/repeater suitable for use in PCIe Gen 1, 2, and 3 applications. These demonstrations were first publicly displayed at the DesignCon conference in Santa Clara, California during the first week of February, 2010. Since the VSC3308 8 x 8 crosspoint switch is identical in performance to the VSC3316, all information in this application note applies equally to that device.

2 PCI Express Repeater Requirements

The concept of a PCIe buffer/repeater is not part of the PCIe standards and specifications. PCIe standards are written around a well defined worst-case channel and the specifications are intended to ensure that compliant devices will never need a buffer/repeater as long as the channel does not exceed the worst case. Since there are no PCIe standards for a buffer/repeater, it is misleading to refer to such a device as “PCIe compliant.” However, careful study of the PCIe architecture and protocol shows that the following functional requirements are necessary for a buffer/repeater device to be fully compatible with PCIe.

2.1 Spread-Spectrum Clocking

PCIe supports optional spread-spectrum clocking used to reduce electromagnetic interference (EMI). “The data rate may be modulated from +0% to -0.5% of the nominal data rate frequency, at a modulation rate in the range not exceeding 30 kHz – 33 kHz.”

A buffer/repeater would be required to support spread-spectrum clocking unless its use is limited to PCIe systems that do not use this option.

2.2 Beacon

The PCIe base specification states that beacon is “An optional 30kHz - 500MHz in-band signal used to exit the L2 Link power management state” and “Support for Beacon is required for all PCI Express components that support a wakeup mechanism in order to function in form factors that require the use of Beacon. “

Unless it is restricted to PCIe systems that do not support beacon, a buffer/repeater would be required to be able to detect beacon signals at the inputs and pass them on to the outputs.

2.3 Electrical Idle

Electrical idle is “The state of the output driver in which both lines, D+ and D-, are driven to the DC common mode voltage.” Electrical idle is used throughout PCI express. A buffer/repeater must have the ability to detect electrical idle at its inputs (e.g., loss of signal) and pass this condition on to the outputs (e.g., mute the outputs to the common mode).

2.4 Receiver Detect

PCIe is a point-to-point interface (i.e., one transceiver communicating with only one other transceiver). The communications link between PCIe devices can be composed of 1 to 32 independent lanes. Any PCIe device that is capable of using more than one lane must be able to communicate with other PCIe devices that may only be capable of using fewer lanes. When PCIe devices initially establish a link, one of the first tasks is to determine how many lanes are supported by the device at the other end of the link. This is accomplished by using the PCIe receiver detect function.

According to the PCIe base specification, “The Receiver Detection circuit is implemented as part of a transmitter and must correctly detect whether a load impedance equivalent to a DC impedance implied by the ZRX-DC parameter (40 Ω - 60 Ω) or lower is present.” Receiver detection occurs when the transmitters on each available lane send a common-mode voltage pulse down their respective lanes and measure the time it takes for the outputs to change to the new voltage level. If a receiver is present on the other end of the lane, the nominal 50-ohm input termination will cause the output voltage to change at a different rate than if there was no 50-ohm termination. By performing the receiver detection on each of its lanes, the PCI device determines how many of its lanes correspond to active receivers on the other end of the link.

In proprietary systems where the number of lanes on each side of the link is known, a buffer/repeater would not need to implement receiver detect. However, for a general purpose PCIe compatible buffer/repeater, implementation of the PCIe receiver detect function is required.

2.5 Dynamic Rate Negotiation

First generation PCIe devices are capable of transmitting and receiving 2.5 Gbps on each lane. PCIe Gen 2 is 5.0 Gbps on each lane and PCIe Gen 3 is 8.0 Gbps on each lane. Regardless of the generation, however, all PCIe devices are required to be

backward compatible with previous generations (e.g., a Gen 3 device must be able to communicate with a Gen 1 or Gen 2 device). When two PCIe devices are initially connected together through a link, they do not know if the device at the other end is capable of any data rate other than the Gen 1 rate of 2.5 Gbps. Therefore all PCIe devices initially communicate at 2.5 Gbps and then exchange information about their respective maximum data rate capabilities. After the link is established at 2.5 Gbps then, if the transceivers on both ends are capable of higher rates, the rate may be changed to 5.0 Gbps or 8.0 Gbps. If, after the link is established at the highest common data rate, there is an interruption or degradation in the channel, the transceivers will drop back down to a lower rate.

For the reasons outlined above, a PCIe Gen 3 buffer/repeater must be able to operate at 2.5 Gbps, 5.0 Gbps, and 8.0 Gbps. In addition, the buffer/repeater must be able to recognize changes in data rates and re-synchronize within the Fast Training Sequence (FTS) time frame allowed by the PCIe specification (10,200 bits for Gen 1 and Gen 2, and 34,190 bits for Gen 3).

2.6 Transmitter De-Emphasis

PCIe Gen 1 transmitters are required to use a fixed -3.5dB de-emphasis for all transmitted signals. For PCIe Gen 2 the de-emphasis is initially set to -3.5dB, but can be increased to -6dB as needed. For PCIe Gen 3 the de-emphasis is determined by a set of three coefficients representing the pre-cursor, cursor, and post-cursor values of the de-emphasis, and these coefficients can be changed as needed when the 8.0 Gbps data rate is initialized.

A PCIe Gen 3 buffer/repeater must be able to function in a link that includes dynamically changing de-emphasis levels.

2.7 PCIe Gen 3 Dynamic Equalization Negotiation

When PCIe Gen 3 transmitters and receivers are establishing an 8.0 Gbps link, they follow a four-phase equalization procedure. Under control of the receiver, the transmitter de-emphasis coefficients are changed and the receiver input equalization is adjusted for each lane in the link in order to find the optimum de-emphasis and equalization combination and guarantee a bit error ratio (BER) of 10^{-12} or better. Depending on the architecture of the buffer/repeater, this can cause difficulties.

A PCIe Gen 3 buffer/repeater must be able to function and maintain link integrity as the link equalization parameters are being adjusted by the PCIe receiver.

3 Meeting the Requirements for a PCIe Gen 3 Buffer/Repeater

The VSC3308 and VSC3316 are asynchronous crosspoint switches capable of data rates up to 11.5 Gbps. They can be used not only as a switch, but also as a buffer that provides signal equalization on inputs and outputs to reverse signal degradations caused by losses in the transmission path.

The VSC3308 has 8 differential inputs and 8 differential outputs, while the VSC3316 has 16 differential inputs and 16 differential outputs. Other than the numbers of inputs and outputs, the devices are architecturally the same. Writing to internal registers (via an external 4-wire SPI or 2-wire I^2C interface) controls the input-to-output connections, with any input capable of being connected to any output. Each input features programmable signal equalization capability of up to 26 dB and each output can be programmed to provide up to 9dB of pre-emphasis.

The fact that these devices are asynchronous means that they do not contain phase-locked loops (PLLs) or clock/data recovery (CDR) circuits that must acquire phase lock with the data signals. Any data rate from zero to 11.5 Gbps is transferred directly from the input to the output of the device with very low latency (~500ps) and the device does not know or care what the data rate is. This is a distinct simplicity advantage when designing for PCIe requirements that involve changing data rates, such as spread-spectrum clocking, beacon, and dynamic rate negotiation.

Equalization capabilities of the VSC3316 are documented for backplanes and cables in two separate Vitesse application notes that are available from the Vitesse web site on the VSC3316 product page:

- "Equalizing 28AWG Cables with the VSC3316" (http://www.vitesse.com/products/download.php?fid=4304&number=VSC3316) shows that the VSC3316 can equalize 36 meters of 28AWG cable at 2.5 Gbps, 17 meters at 5.0 Gbps, and 10 meters at 8.0 Gbps.
- "Equalizing Backplanes with the VSC3316" (http://www.vitesse.com/products/download.php?fid=4311&number=VSC3316) shows that the VSC3316 can equalize 120 inches of FR-4 backplane at 2.5 Gbps, 90 inches at 5.0 Gbps, and 30 inches at 8.0 Gbps.

3.1 Spread-Spectrum Clocking, Beacon, and Dynamic Rate Negotiation

As noted above, the asynchronous nature of the VSC3308 and VSC3316 means that they are not affected by changes in clock rates or data rates. This means that spread-spectrum clocking, beacon, and dynamic rate negotiation are completely transparent to the devices.

3.2 Electrical Idle

The VSC3308 and VSC3316 detect the PCIe electrical idle state at their inputs using their loss of signal (LOS) circuits. Writing to the LOS registers via the external 4-wire SPI or 2-wire I^2C interfaces sets the LOS threshold, and the "output mode" register can be set up to mute the corresponding outputs to the common-mode voltage when LOS is detected. Using these settings makes the VSC3308 and VSC3316 fully compliant with PCIe electrical idle.

3.3 Receiver Detect

The VSC3308 and VSC3316 feature independently-adjustable common-mode impedance at each input and separate receiver detect circuits on each output. When

combined with a simple external controller, the VSC3308 and VSC3316 can be set up to be fully compliant with the PCIe receiver detect function.

Since the buffer/repeater is physically located between the PCIe transmitter and receiver, it must transparently relay receiver detect information from its outputs to its inputs, so that the PCIe transmitters reliably detect which lanes are connected to active receivers. This is accomplished by writing to the VSC3308/VSC3316 receiver detect register of an output, which causes the output to: (1) mute to the common-mode, (2) send out a common-mode pulse, and (3) log the output voltage rise time. The presence or absence of a PCIe receiver connected to the output is inferred based on the output voltage rise time, which is stored in a register on the VSC3308/VSC3316. An external controller is needed to read the output voltage rise time from the register and, if the rise time indicates that a receiver is present, the controller sets the VSC3308/VSC3316 input common-mode impedance low. Otherwise, if no receiver is present, the input common-mode impedance is set high.

The above method of accomplishing receiver detect has been fully implemented and successfully tested using a VSC3316 EV board. The stock VSC3316 EV board includes a SiLabs C8051F311 microcontroller that is used to facilitate communications between an external USB interface and the on-board I^2C bus. By adding additional code to the existing microcontroller flash memory, the receiver detect function was implemented without adding any hardware. The source code for the receiver detect control function is listed as an appendix to this application note and available in electronic form upon request.

An outline of the microcontroller receiver detect functionality is as follows:

(A) Check status of input LOS (Channel Status Register, bit 0)

- If loss of signal, then electrical idle is indicated. Proceed to (B)
  else return to (A)

(B) Initiate receiver detect on the output (write ‘1’ to Output Mode Register, bit 7)

(C) Read the receiver detect results (Channel Status Register, bit 2 and bit 1)

- If receiver detected, set input common-mode impedance low (Input State Register, bit 1 = 0)
  else set input common-mode impedance high (Input State Register, bit 1 = 1)

(D) Go back to (A)

3.4 Transmitter De-emphasis

The VSC3308/VSC3316 devices include adjustable pre-emphasis on all outputs. Available pre-emphasis levels include PCIe Gen 1 and 2 required levels of +3.5 and +6.0 dB. By using the adjustable output amplitude to decrease the output swing as the pre-emphasis is increased, the result is the equivalent of de-emphasis at -3.5 and -6.0 dB.

As noted in sections 2.6 and 2.7, the PCIe de-emphasis requirements for Gen 3 are significantly different than Gen 1 and Gen 2. In Gen 3 the de-emphasis settings have a wider range of possibilities and the particular de-emphasis setting that is used is determined by the receiver at the other end of the lane as part of the dynamic

equalization process. The method used by the VSC3308/VSC3316 to function with PCIe Gen 3 de-emphasis requirements is detailed in the following section.

3.5 PCIe Gen 3 Dynamic Equalization Negotiation

PCI express is designed specifically for two channel models: the “client channel” which has one connector and 14 inches of FR-4 microstrip, and the “server channel” with two connectors and 20 inches of FR4 stripline. Based on the specified worst-case channel characteristics, PCIe compliant transmitters and receivers are designed to achieve a bit error ratio of 10^{-12} or better over any PCIe compliant channel.

As the data rate is increased to 8.0 Gbps in PCIe Gen 3, the complexity of the transmit and receive equalization increases dramatically in order to maintain a BER of 10^{-12} over the same worst-case channel models that were used for Gen 1 and Gen 2. In Gen 3, the transmitter is required to adjust its output de-emphasis using a set of three coefficients. These coefficients represent the pre-cursor, cursor, and post-cursor values of the de-emphasis, and the coefficients can be changed as needed when the 8.0 Gbps data rate is initialized in order to reach the 10^{-12} goal for BER. A Gen 3 receiver is designed to use techniques such as adjustable continuous time linear equalization (CTLE) and distributed feedback equalization (DFE).

During 8.0 Gbps link initialization, the four-phase dynamic equalization process is controlled by the receivers at the ends of each lane in the link. The receivers adjust their input equalization settings and send de-emphasis coefficient values to the transmitters on the other end of their lanes. Transmitters are required to comply with requests for de-emphasis coefficient changes. The receiver determines the best combination of transmitter de-emphasis and input equalization based on optimization of the eye diagram opening. After the best equalization settings are found, they are held static and the dynamic equalization process is exited (i.e., there is no continuing adaptive equalization after the link has been successfully initialized).

Two key assumptions in the PCIe dynamic equalization process are: (1) the BER is guaranteed to be no worse than 10^{-4} (with the least optimum equalization settings and worst-case channel), and (2) the BER is guaranteed to be better than 10^{-12} for at least one combination of equalization settings and the worst-case channel. The first assumption is necessary because the initial communication between the receiver and transmitter during the equalization negotiation process (e.g., de-emphasis coefficient change requests from the receiver to the transmitter) is designed to occur over the non-optimized channel. Redundancy, error detection/correction, etc., that are used over the non-optimized channel are designed for a worst-case BER of 10^{-4}. If the BER is worse the 10^{-4} at the start of the dynamic equalization process, it may be impossible for the receiver to communicate de-emphasis coefficient change requests to the transmitter, in which case 8.0 Gbps link initialization will fail, and the system will be forced to drop back to a slower data rate. The second assumption (that a BER better than 10^{-12} is possible with the optimum combination of transmitter and receiver equalization settings) is only valid with a PCIe compliant channel. If the channel characteristics are beyond the ranges defined for PCIe, then both of the BER assumptions become invalid.

For channels that are degraded beyond the PCIe defined ranges, the PCIe Gen 3 process of dynamic equalization negotiation necessary for 8.0 Gbps data rates may fail. The purpose of the PCIe Gen 3 buffer/repeater is to improve the degraded channel enough to ensure successful equalization negotiation and sustained operation at 8.0 Gbps. Once the buffer/repeater has improved the channel enough for the PCIe transmitters and receivers to communicate with a BER of at least 10^{-4}, the PCIe Gen 3

dynamic equalization process can begin and the buffer/repeater should keep its equalization setting static (i.e., if the buffer/repeater used continuously adaptive equalization, it might interfere with the PCIe dynamic equalization negotiation).

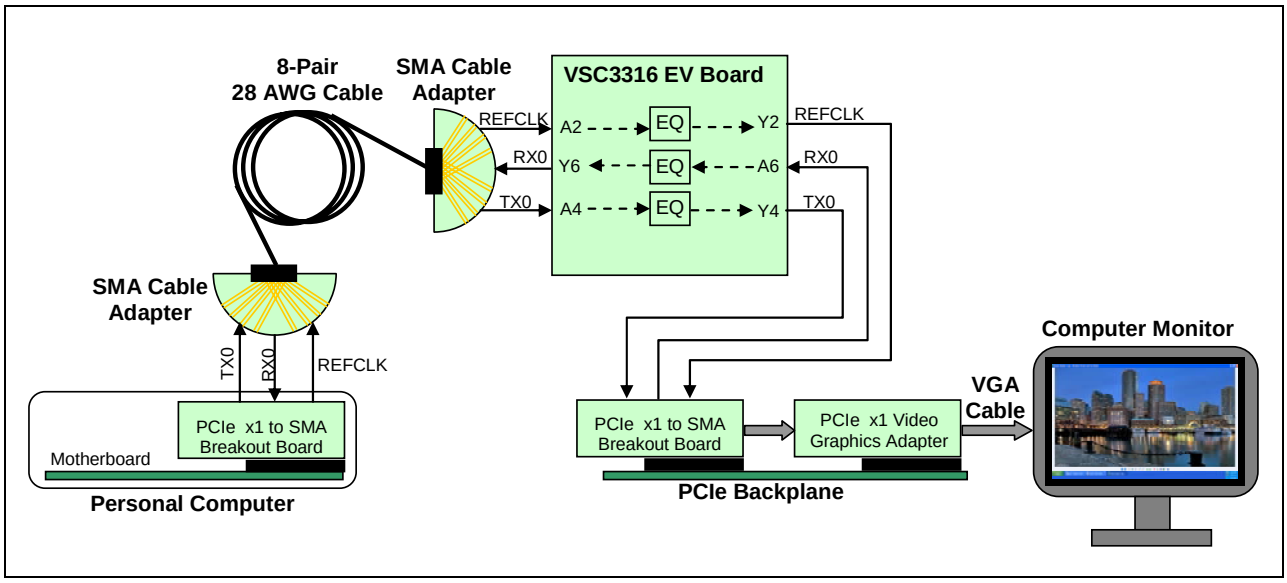
A potential complication is that, during PCIe dynamic equalization negotiation, the PCIe receiver will be sending requests for de-emphasis coefficient changes to the PCIe transmitter, but the resulting de-emphasis changes may be fully or partially blocked by the buffer/receiver. As the PCIe transmitter responds to the de-emphasis change requests, the de-emphasis applied to the transmitted signals will reach the buffer/repeater, but may not propagate through the buffer/repeater to the PCIe receiver. This is the case with the VSC3308/VSC3316 and therefore de-emphasis adjustments by the PCIe transmitter may cause improvement or degradation of the signal received by the buffer/repeater. The improving or degrading signal quality at the buffer/repeater input can either: (1) translate to corresponding improvement or degradation of the signal transmitted by the buffer/repeater to the PCIe receiver, or (2) result in no noticeable effect on the signal transmitted by the buffer/repeater to the PCIe receiver. In the first case, the PCIe receiver will detect changes in received signal quality in response to de-emphasis coefficient change requests, causing the PCIe receiver to settle on the optimum set of de-emphasis coefficients. In the second case, the PCIe receiver will not detect any change in signal quality in response to de-emphasis coefficient changes and the receiver will stop requesting coefficient changes when the time-out period expires. Either way, the probability of achieving stable 8.0 Gbps operation at a BER better than 10^{-12} will be enhanced (the probability of stable 8.0 Gbps operation without the repeater/buffer would tend toward zero). If the channel degradation is beyond the capability of the buffer/repeater equalizer at 8.0 Gbps, then PCIe protocol will remain stable and the link data rate will be decreased to 5.0 Gbps or 2.5 Gbps in the normal manner.

4 PCIe Demonstration of the VSC3316

At the time of this writing, PCIe Gen 3 compliant devices are not yet available. For the reasons outlined above, however, Vitesse is highly confident that the VSC3308/VSC3316 will function well in PCIe Gen 3 applications. The VSC3308 and VSC3316 are PCIe Gen 3 ready.

In order to fully demonstrate operation of the VSC3316 with existing PCIe devices, PCIe Gen 1 and Gen 2 were targeted. The test setup is illustrated in Figure 1.

Figure 1. PCIe Test Setup with the VSC3316



This test setup incorporated a standard Dell desktop PC that includes a PCIe x16 connector to interface with a PCIe x1 video adapter card. The video adapter was removed and replaced with the PCIe to SMA adapter board as illustrated in Figure 2 and Figure 3.

Figure 2. PCIe to SMA Adapter

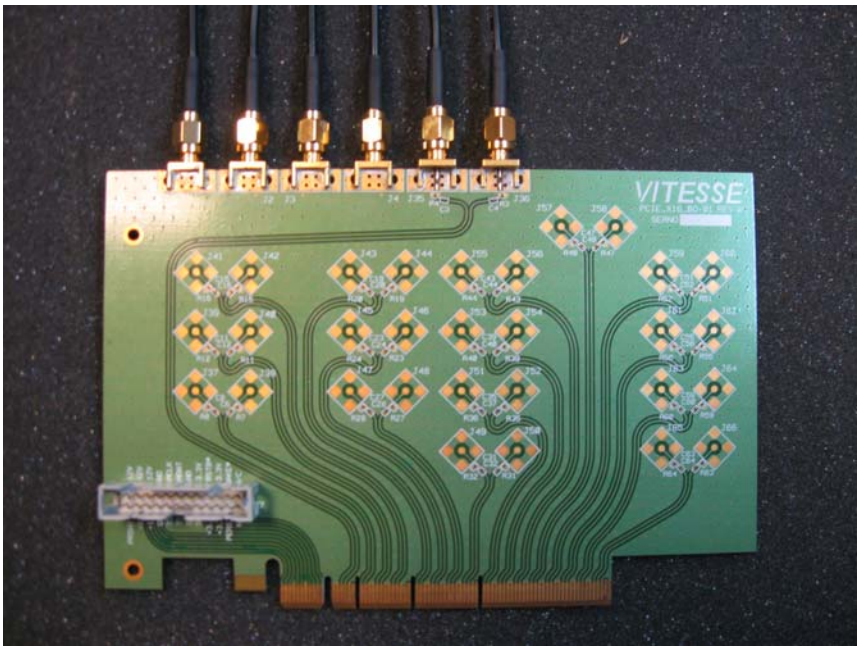
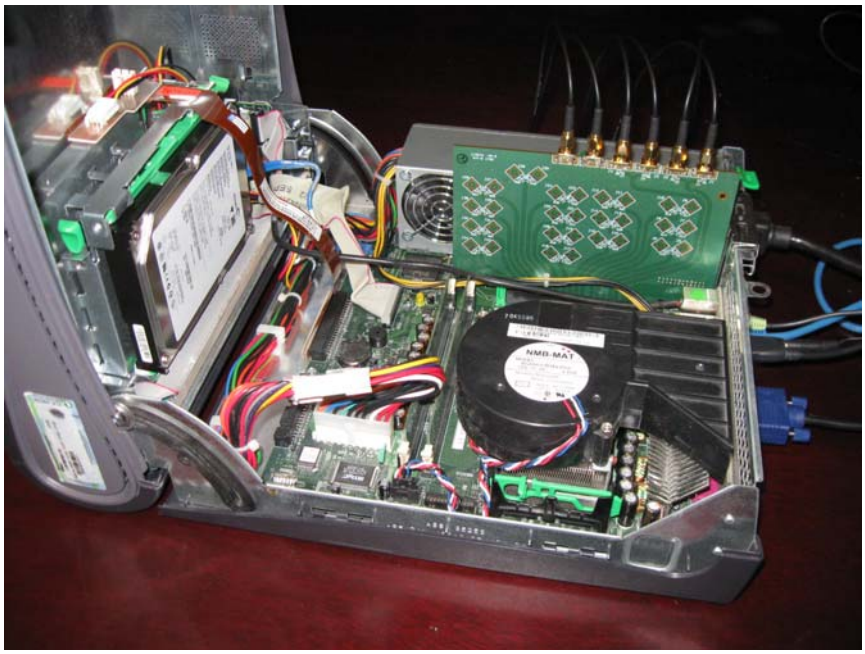
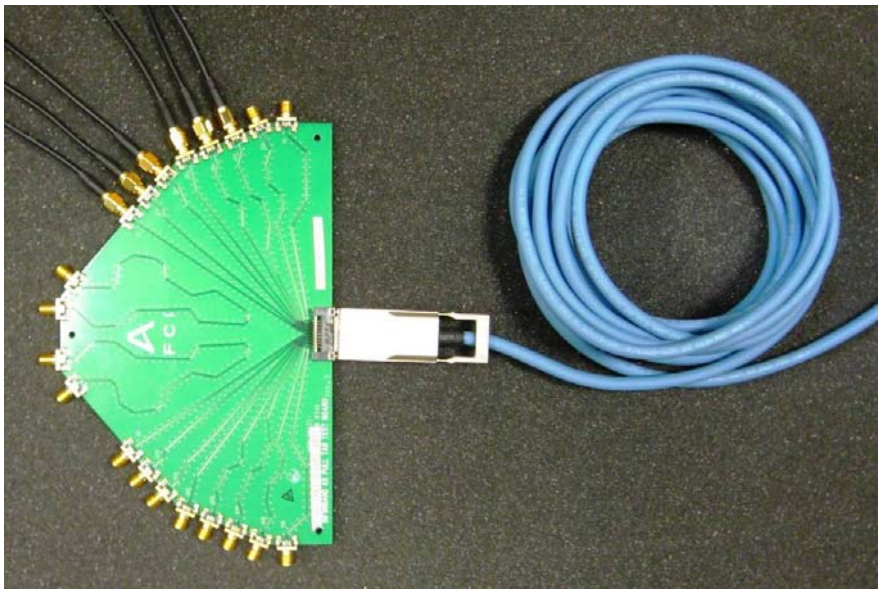


Figure 3. Desktop PC with PCIe to SMA Adapter



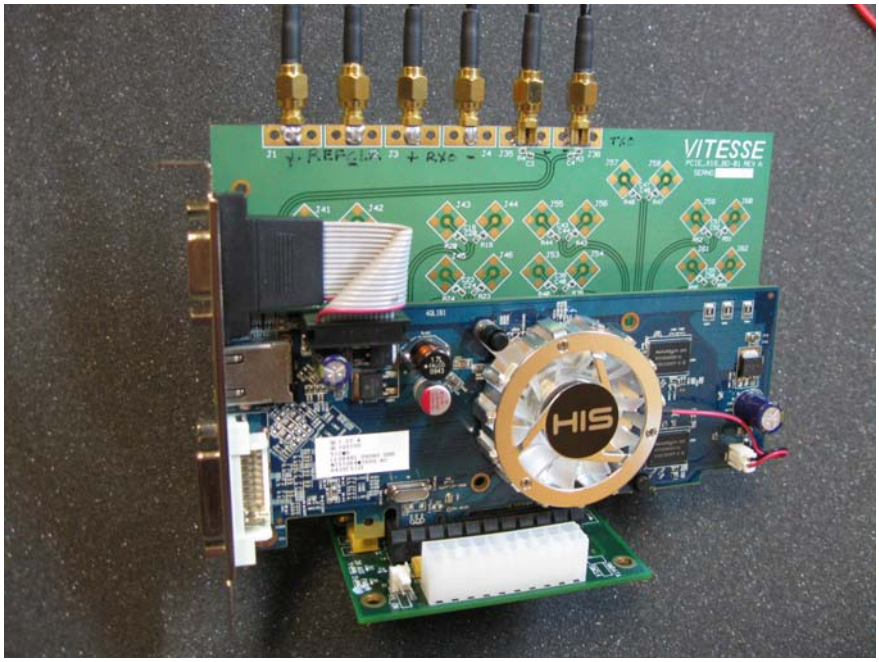
Next, the PCIe RX0, PCIe TX0, and PCIe reference clock were connected from the SMA adapter board to a 5 meter 28AWG cable using the cable adapter shown in Figure 4. This provides a much longer channel than the PCIe standard.

Figure 4. SMA to 28AWG Cable Adapter



On the other end of the 28AWG cable, another SMA to 28AWG cable adapter was used with another set of SMA cables to connect with the video adapter card using a mini-PCIe 2-slot backplane as shown in Figure 5.

Figure 5. 2-Slot PCIe Backplane Connecting to Video Card



Finally, the video adapter card was connected to a computer monitor using a standard VGA cable. The result was a perfect picture in the computer monitor – exactly the same as when the video adapter was plugged directly in to the desktop PC.

The success of this demonstration shows that the VSC3316 supports all of the PCIe repeater requirements listed in section 2, and that it can extend the PCIe channel far beyond the personal computer.

5 Conclusions

PCI Express specifications concentrate on a well-defined range of channel characteristics. If the channel in a system is degraded beyond the PCIe specified range, there is no guarantee that PCIe compliant devices by themselves will be able to successfully establish a link. In this case a PCIe buffer or PCIe repeater can be used to compensate for the channel degradation so that standard PCIe devices can communicate.

A successful PCIe buffer/repeater device must do more than meet the data rate requirements of PCIe (2.5 Gbps for Gen 1, 5.0 Gbps for Gen 2, and 8.0 Gbps for Gen 3). PCIe buffer/repeater devices must also be compatible with lower frequency requirements of the PCIe protocol, such as beacon, electrical idle, and receiver detect while at the same time dealing with changing data rates. The advent of PCIe Gen 3 is now bringing a whole new set of dynamic equalization requirements that must also be comprehended by the PCIe buffer/repeater.

This application note documents the requirements for a successful PCIe buffer/repeater and shows how the VSC3308 and VSC3316 meet these requirements. Demonstration of the VSC3316 in a commercially available PCIe system proves that it is capable of all of the necessary PCIe-unique protocol requirements. Separate demonstrations of the VSC3316 at data rates from 2.5 to 11.5 Gbps over long cables and backplanes prove the equalization benefits of the VSC3308 and VSC3316.

The VSC3308 and VSC3316 are proven excellent PCIe buffer/repeaters for PCIe Gen 1 and Gen 2, and they are ready for Gen 3.

6 APPENDIX: Source Code for Receiver Detect Firmware

```
//=============================================================================================

//  =   Copyright(c) Vitesse Semiconductor Corporation

//  =

//  =   This software has been provided "as is," without express or implied warranty including,
without limitation, implied warranties of merchantability, fitness for

//  =   a particular use and non-infringement.

//
=============================================================================================

//  =   Project     PCIe with VSC3316

//  =   File        vsc.c

//  =

//
=============================================================================================

#define             VSC_GLOBALS

#include            "includes.h"

//
=============================================================================================

//  =   Local constants and variables

//
=============================================================================================

#define             VSC_IDLE            0

#define             VSC_WAIT_50MS       1

#define             VSC_HYSTERESIS      2


u8 xdata            VSC_InputState[16];

u8 xdata            VSC_Fsm[16];

u8 xdata            VSC_OutputMode[16];

u8 xdata            VSC_Timeout[16];

u8                  VSC_Interface;


//
=============================================================================================
```

```
//  =   Global functions
//
//============================================================================================
//
//============================================================================================
//  =   Function    VSC_Init3316
//  =
//
//============================================================================================
void                VSC_Init3316(void) {


  if (VSC_Interface == CONFIG_DEFAULT) {
    I2C_InitVsc3316();
  }
  else {
    SPI_InitVsc3316();
  }
}
//
//============================================================================================
//  =   Function    VSC_InitFsm
//  =
//
//============================================================================================
void                VSC_InitFsm(void) {
u8                  i;


  VSC_Interface = CBYTE [CONFIG_3316_IF];                                    //
  VSC_Fsm_Control = 1;                                                       //

//____________________________________________________________________________


  for (i = 0; i < 16; i++) {                                                 //
    VSC_Fsm[i] = VSC_IDLE;                                                   //
    VSC_Timeout[i] = 0;                                                      //
```

```
  }
//_____________________________________________________________________________________


  for (i = 0; i < 16; i++) {                          //  get information from Flash configuration
                                                          for registers in page 0x13

    VSC_InputState[i] = CBYTE [VSC3316_CONFIG + ( 4 * 18) + i];        //  page 0x13

  }
//_____________________________________________________________________________________


  for (i = 0; i < 16; i++) {                          //  get information from Flash configuration
                                                          for registers in page 0x23

    VSC_OutputMode[i] = CBYTE [VSC3316_CONFIG + (10 * 18) + i];        //  page 0x23

  }
//_____________________________________________________________________________________

}


//
=======================================================================================

//  =   Function    VSC_Read3316

//  =

//
=======================================================================================


u8                  VSC_Read3316(u8 addr) {

u8                  databyte;


  if (VSC_Interface == CONFIG_DEFAULT) {

    databyte = I2C_ReadVsc3316(addr);

  }

  else {

    databyte = SPI_ReadVsc3316(addr);

  }

  return databyte;

}


//
=======================================================================================
```

```
//  =   Function    VSC_ReceiverDetect
//  =
//
================================================================================================


void                VSC_ReceiverDetect(void) {
u8                  i;
u8                  currentpage;


  if (VSC_Fsm_Control) {
    currentpage = VSC_Read3316(0x7F);                       //  save current page


    for (i = 0; i < 16; i++) {                              //  check all 16 channels
      if ((VSC_InputState[i] & 0x01) == 0) {                //  if ChannelInputPower is supposed to
                                                                be ON, perform ReceiverDetect
        switch (VSC_Fsm[i]) {                               //
                                                            //
        case VSC_IDLE:                                      //  - VSC_IDLE
          VSC_Write3316(0x7F, 0xF0);                        //  - get ChannelStatus
          if ((VSC_Read3316(i) & 0x01) == 0x01) {           //  - if ChannelLosStatus
            VSC_Write3316(0x7F, 0x23);                      //  - start ReceiverDetect **********
                                                                ok for now in_i connect to out_i,

            VSC_Write3316(i, (VSC_OutputMode[i] | 0x80));       //
            VSC_Timeout[i] =  ((u8) (INIT_SysTick + 56)) & 0xFC;   //  - set timeout
            VSC_Fsm[i] = VSC_WAIT_50MS;                         //  - goto VSC_WAIT_50MS
          }                                                     //
          break;                                                //
                                                                //
        case VSC_WAIT_50MS:                                    // - VSC_WAIT_50MS
          if ((INIT_SysTick & 0xFC) ==  VSC_Timeout[i]) {       //  - if timeout
            VSC_Write3316(0x7F, 0xF0);                          //   - read ReceiverDetect result:
            if ((VSC_Read3316(i) & 0x06) == 0x04) {             //    - ReceiverPresent
              VSC_Write3316(0x7F, 0x13);                        //          keep termination at VDD
              VSC_Write3316(i, (VSC_InputState[i]       ));      //
```

```
            }                                                       //
            else {                                                  //  - NoReceiver / Error
              VSC_Write3316(0x7F, 0x13);                            //  change termination to C_INTERM
              VSC_Write3316(i, (VSC_InputState[i] | 0x02));         //
            }                                                       //
            VSC_Write3316(0x7F, 0x23);                            //  - stop ReceiverDetect
            VSC_Write3316(i, VSC_OutputMode[i]);                     //
            VSC_Timeout[i] =  ((u8) (INIT_SysTick + 56)) & 0xFC;    //  - set timeout
            VSC_Fsm[i] = VSC_HYSTERESIS;                             //  - goto VSC_HYSTERESIS
          }                                                          //
          break;                                                     //
                                                                     //
        case VSC_HYSTERESIS:                                         //  - VSC_HYSTERESIS
          if ((INIT_SysTick & 0xFC) ==  VSC_Timeout[i]) {            //   - if timeout
            VSC_Fsm[i] = VSC_IDLE;                                   //    - goto VSC_IDLE
          }                                                          //
          break;                                                     //
                                                                     //
        }                                                            //
      }                                                              //
    }                                                                //

    VSC_Write3316(0x7F, currentpage);                                //  restore current page
  }
}
```

```
//
//============================================================================================
//  =   Function    VSC_Write3316
//  =
//
//============================================================================================


void                VSC_Write3316(u8 addr, u8 databyte) {


  if (VSC_Interface == CONFIG_DEFAULT) {
    I2C_WriteVsc3316(addr, databyte);
  }
  else {
    SPI_WriteVsc3316(addr, databyte);
  }
}
```