Using LVDS I/O

Introduction

Low Voltage Differential Signaling (LVDS) is a very popular and powerful high-speed interface in many system applications. Virtex-II I/Os are designed to comply with the IEEE electrical specifications for LVDS to make system and board design easier. With the addition of an LVDS current-mode driver in the IOBs, which eliminates the need for external source termination in point-to-point applications, and with the choice of two different voltage modes and an extended mode, Virtex-II devices provide the most flexible solution for doing an LVDS design in an FPGA.

Table 2-56 lists all LVDS primitives that are available for Virtex-II devices.

Table 2-56: **Available Virtex-II LVDS Primitives**

Input	Output	3-State	Clock	Bi-Directional
IBUF_LVDS	**OBUF_LVDS**	**OBUFT_LVDS**	**IBUFG_LVDS**	**IOBUF_LVDS**
IBUFDS_LVDS_25	OBUFDS_LVDS_25	OBUFTDS_LVDS_25	IBUFGDS_LVDS_25	
IBUFDS_LVDS_33	OBUFDS_LVDS_33	OBUFTDS_LVDS_33	IBUFGDS_LVDS_33	
IBUFDS_LVDSEXT_25	OBUFDS_LVDSEXT_25	OBUFTDS_LVDSEXT_25	IBUFGDS_LVDSEXT_25	
IBUFDS_LVDSEXT_33	OBUFDS_LVDSEXT_33	OBUFTDS_LVDSEXT_33	IBUFGDS_LVDSEXT_33	

The primitives in **bold** type are pre-existing LVDS primitives used in Virtex-E and earlier designs. They are not current-mode drivers and are still required for BLVDS (Bus LVDS) applications.

*DS_LVDS_25 = 2.5V V_{CCO} LVDS Buffer

*DS_LVDS_33 = 3.3V V_{CCO} LVDS Buffer

There is no difference in the AC characteristics of either voltage-mode LVDS I/O. These choices now provide more flexibility for mixed-I/O banking rules; that is, an LVTTL I/O can coexist with the 3.3V LVDS buffer in the same bank.

DS_LVDSEXT = Extended mode LVDS buffer

This buffer provides a higher drive capability and voltage swing (350 - 750 mV), which makes it ideal for long-distance or cable LVDS links.

The output AC characteristics of this LVDS driver are not within the EIA/TIA specifications. This LVDS driver is intended for situations that require higher drive capabilities in order to produce an LVDS signal that is within EIA/TIA specification at the receiver.

Creating an LVDS Input/Clock Buffer

Figure 2-108 illustrates the LVDS input and clock buffer primitives shown in Table 2-57. The pin names used are the same as those used in the HDL library primitives.

Table 2-57: **LVDS Input and Clock Buffer Primitives**

LVDS Inputs	LVDS Clocks
IBUFDS_LVDS_25	IBUFGDS_LVDS_25
IBUFDS_LVDS_33	IBUFGDS_LVDS_33
IBUFDS_LVDSEXT_25	IBUFGDS_LVDSEXT_25
IBUFDS_LVDSEXT_33	IBUFGDS_LVDSEXT_33

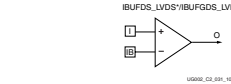

Figure 2-108: **LVDS Input and Clock Primitives**

To create an LVDS input, instantiate the desired mode (2.5 V, 3.3 V, or Extended) LVDS input buffer. Notice that the P and N channels are included in the primitive (I = P, IB = N). Software automatically uses the appropriate pin from an adjacent IOB for the N channel. The same applies to LVDS clocks: Use IBUFGDS_LVDS*

LVDS Input HDL Examples

VHDL Instantiation

```
U1: IBUFDS_LVDS_25
  port map  (
              I => data_in_P,
             IB => data_in_N
              O => data_in
            );
```

Verilog Instantiation

```
IBUFDS_LVDS_25   U1    ( .I(data_in_P),
                         .IB(data_in_N),
                         .O(data_in)
                       );
                );
```

Port Signals

I = P-channel data input to the LVDS input buffer

IB = N-channel data input to the LVDS input buffer

O = Non-differential input data from LVDS input buffer

Location Constraints

```
NET "data_in_P" LOC= "NS";
```

LVDS Receiver Termination

All LVDS receivers require standard termination. Figure 2-109 is an example of a typical termination for an LVDS receiver on a board with 50Ω transmission lines.

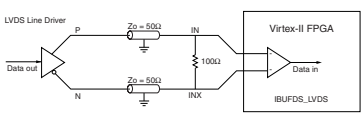

Figure 2-109: **LVDS Receiver Termination**

Creating an LVDS Output Buffer

Figure 2-110 illustrates the LVDS output buffer primitives:

- OBUFDS_LVDS_25
- OBUFDS_LVDS_33
- OBUFDS_LVDSEXT_25
- OBUFDS_LVDSEXT_33

The pin names used are the same as those used in the HDL library primitives.

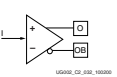

Figure 2-110: **LVDS Output Buffer Primitives**

To create an LVDS output, instantiate the desired mode (2.5, 3.3V, or Extended) LVDS output buffer. Notice that the P and N channels are included in the primitive (O = P, OB = N). Software automatically uses the appropriate pin from an adjacent IOB for the N channel.

LVDS Output HDL Examples

VHDL Instantiation

```
U1: OBUFDS_LVDS_25
  port map  (
              I => data_out,
              O => data_out_P,
              OB => data_out_N
            );
```

Verilog Instantiation

```
OBUFDS_LVDS_25 U1  ( .I(data_out),
                     .O(data_out_P),
                     .OB(data_out_N)
                   );
```

Port Signals

I = data input to the LVDS input buffer

O = P-channel data output

OB = N-channel data output

Location Constraints

```
NET "data_out_P" LOC= "NS";
```

LVDS Transmitter Termination

The Virtex-II LVDS transmitter does not require any termination. Table 2-56 lists primitives that correspond to the Virtex-II LVDS current-mode drivers. Virtex-II LVDS current-mode drivers are a true current source and produce the proper (IEEE/EIA/TIA compliant) LVDS

signal. Figure 2-111 illustrates a Virtex-II LVDS transmitter on a board with 50Ω transmission lines.

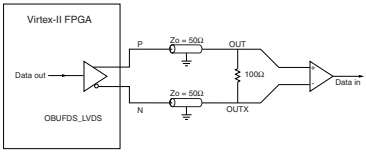

Figure 2-111: **LVDS Transmitter Termination**

Creating an LVDS Output 3-State Buffer

Figure 2-112 illustrates the LVDS 3-State buffer primitives:

- OBUFTDS_LVDS_25
- OBUFTDS_LVDS_33
- OBUFTDS_LVDSEXT_25
- OBUFTDS_LVDSEXT_33

The pin names used are the same as those used in the HDL library primitives.

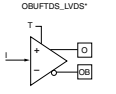

Figure 2-112: **LVDS 3-State Primitives**

To create an LVDS 3-State output, instantiate the desired mode (2.5V, 3.3V, or Extended) LVDS 3-State buffer. Notice that the P and N channels are included in the primitive (O = P, OB = N). Software automatically uses the appropriate pin from an adjacent IOB for the N channel.

LVDS 3-State HDL Example

VHDL Instantiation

```
U1: OBUFTDS_LVDS_25
  port map  (
              I => data_out,
              T => tri,
              O => data_out_P,
              OB => data_out_N
            );
```

Verilog Instantiation

```
OBUFTDS_LVDS_25  U1  ( .I(data_out),
                       .T(tri),
                       .O(data_out_P),
                       .OB(data_out_N)
                     );
```

Port Signals

I = data input to the 3-state output buffer

T = 3-State control signal

O = P-channel data output

OB = N-channel data output

Location Constraints

```
NET "data_out_P" LOC = "NS";
```

LVDS 3-State Termination

The Virtex-II LVDS 3-state buffer does not require any termination. Table 2-56 lists primitives that correspond to the Virtex-II LVDS current-mode drivers. These drivers are a true current source, and they produce the proper (IEEE/EIA/TIA compliant) LVDS signal. The Figure 2-113 illustrates a simple redundant point-to-point LVDS solution with two LVDS 3-state transmitters sharing a bus to one LVDS receiver and the required termination for the circuit.

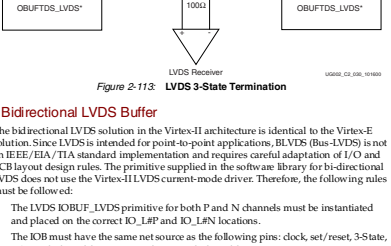

Figure 2-113: **LVDS 3-State Termination**

Creating a Bidirectional LVDS Buffer

The bidirectional LVDS solution in the Virtex-II architecture is identical to the Virtex-E solution. Since LVDS is intended for point-to-point applications, BLVDS (Bus-LVDS) is not an IEEE/EIA/TIA standard implementation and requires careful adaptation of I/O and PCB layout design rules. The primitive supplied in the software library for bi-directional LVDS does not use the Virtex-II LVDS current-mode driver. Therefore, the following rules must be followed:

- The LVDS IOBUF_LVDS primitive for both P and N channels must be instantiated and placed on the correct IO_L#P and IO_L#N locations.
- The IOB must have the same net source as the following pins: clock, set/reset, 3-State, 3-State clock enable, output, and output clock enable.

- The output pins must be inverted with respect to each other, and if output registers are used, the INIT states must be opposite values.
- If 3-State registers are used, they must be initialized to the same state.

Failure to follow these rules results in DRC errors in software.

VHDL Instantiation

```
P_Channel: IOBUF_LVDS
  port map (
                I => data_out,
                T => tri,
                IO=> data_out_P,
                O => data_int
                );
Data_Inv: INV
  port map (
                I => data_out,
                O => data_out_inv
                );
N_Channel: IOBUF_LVDS
 port map (
                I => data_out_inv,
                T => tri,
                IO=> data_out_N,
                O => open
               );
```

Verilog Instantiation

```
IOBUF_LVDS  P_Channel     (.I(data_out),
                           .T(tri),
                           .IO(data_out_P),
                           .O(data_int)
                           );

INV Data_Inv  ( .I(data_out),
                .O(data_out_inv)
                );

IOBUF_LVDS  P_Channel   (.I(data_out_inv),
                         .T(tri),
                         .IO(data_out_N),
                         .O()
                        );
```

Port Signals

I = data output: internal logic to LVDS I/O buffer.

T = 3-State control to LVDS I/O buffer.

IO = Data output from LVDS I/O buffer.

O = Data input: off-chip data to LVDS I/O buffer

Location Constraints

The IOBUF_LVDS buffers must be explicitly placed on a device. This can be accomplished by adding the following constraints in the .ucf or .ncf files.

```
NET data_out_P LOC = D28; #IO_L0P
NET data_out_N LOC = B29; #IO_L0N
```

LDT

Lightning Data Transport (LDT) is a new high speed interface and protocol introduced by Advanced Micro Devices. LDT is a differential signaling based interface that is very similar to LVDS. Virtex-II IOBs are equipped with LDT buffers. These buffers also have corresponding software primitives as follows:

- IBUFDS_LDT_25
- IBUFGDS_LDT_25
- OBUFDS_LDT_25
- OBUFTDS_LDT_25

LDT Implementation

LDT implementation is the same as LVDS with DDR, so follow all of the rules and guidelines set forth earlier in this chapter for LVDS-DDR, and replace the LVDS buffer with the corresponding LDT buffer. For more information on Virtex-II LDT electrical specification, refer to the *Virtex-II Data Sheet*.