White Paper

DPA Circuitry and rx_dpa_locked Signal Behavior in Stratix III Devices

Introduction

The receiver PLL provides eight clock phases to the DPA circuitry. The eight clock phases are separated by 45° and at a frequency equal to the serial data rate. After power up or reset, the DPA circuitry selects an optimum clock phase out of these eight clock phases to sample the received data. The DPA circuitry does not require a fixed training pattern to select the optimum phase. After power up or reset, the DPA circuitry relies on transitions on the received data to select the optimum phase.

ALTLVDS Megafunction

The ALTLVDS megafunction provides an optional output signal `rx_dpa_locked` to indicate when the DPA has locked to the optimum phase. In Stratix® III devices, the logic that drives the `rx_dpa_locked` signal is implemented in the FPGA fabric and is automatically instantiated on a per-channel basis along with the ALTLVDS instance. Once the DPA locks to the optimum phase, the `rx_dpa_locked` signal always stays high until you assert the `rx_reset` signal of the associated LVDS channel or the `pll_areset` signal of the receiver PLL providing the eight clock phases to the DPA circuitry.

☞ The `rx_dpa_locked` signal is no longer driven by the hard macro in Stratix III devices. The `rx_dpa_locked` signal only indicates an initial DPA lock condition to the optimum phase after power up or reset. The `rx_dpa_locked` signal does not get de-asserted if the DPA selects a new phase out of the eight clock phases to sample the received data. You must not use the `rx_dpa_locked` signal to determine a DPA Loss of Lock condition.

Altera recommends:

- Resetting the synchronizer (FIFO buffer) using the `rx_fifo_reset` signal once after the `rx_dpa_locked` signal gets asserted and before valid data is received.
- Using error checkers; for example, Cyclic Redundancy Check (CRC) or Diagonal Interleaved Parity (DIP-4), to validate the integrity of the LVDS link.

Configuring the ALTLVDS Megafunction in DPA Mode

The receiver ALTLVDS MegaWizard® Plug-In Manager provides a **Use External PLL** option to allow more flexible settings for the receiver PLL. If this option is disabled, the receiver PLL is automatically instantiated as a part of ALTLVDS instance. If this option is enabled, you must use the ALTPLL MegaWizard Plug-In Manager to instantiate the receiver PLL for the LVDS link.

The behavior of the `rx_dpa_locked` signal is same with or without the **Use External PLL** option enabled.

The logic that drives the `rx_dpa_locked` signal needs inputs from the receiver PLL and the LVDS receiver channel. If the **Use External PLL** option is disabled, the ALTLVDS instance automatically provides the necessary inputs from the receiver PLL and the LVDS receiver channel to this logic. If the **Use External PLL** option is enabled, you must make the following connections between the ALTPLL and ALTLVDS instances in your design.

WP-01068-1.0

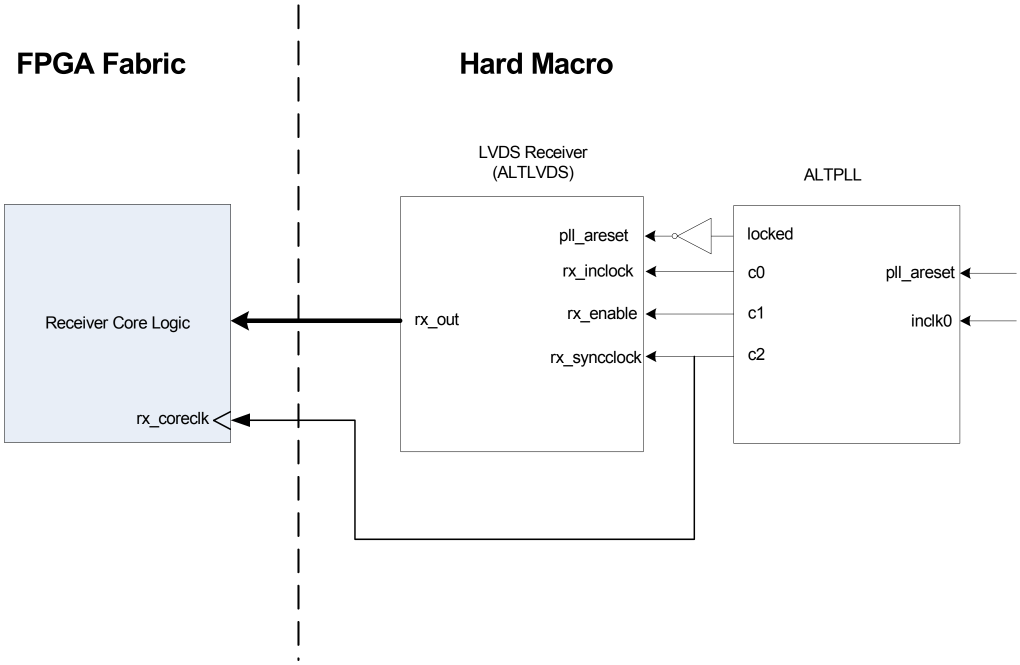
Table 1. Signal Interface Between ALTPLL and ALTLVDS

From ALTPLL (Note 1)	To ALTLVDS Receiver
Serial clock output (c0)	`rx_inclock` (serial clock input)
Load enable output (c1)	`rx_enable` (load enable for the deserializer)
Parallel clock output (c2)	`rx_syncclock` (parallel clock input)
~(locked)	`pll_areset` (asynchronous PLL reset port)

Note to Table 1:
The table shows the serial clock output, load enable output, and parallel clock output generated on ports c0, c1, and c2 of the ALTPLL instance as an example. You can choose any of the PLL output clock ports to generate the above three interface clocks.

Figure 1 illustrates the required ALTPLL-to-ALTLVDS port connections when the ALTLVDS receiver instance is configured in DPA mode with the **Use External PLL** option enabled.

Figure 1. ALTPLL-to-ALTLVDS Port Connections in DPA Mode with the Use External PLL Option Enabled



The two input ports, `rx_inclock` and `rx_enable`, to the LVDS receiver instance are available through the ALTLVDS MegaWizard Plug-In Manager. The ports `pll_areset` and `rx_syncclock` are not available through the ALTLVDS MegaWizard Plug-In Manager and must be manually included by modifying the **.v** or **.vhd** wrapper file generated by the ALTLVDS MegaWizard Plug-In Manager.

Instantiation of `rx_syncclock` in the ALTLVDS wrapper file is mandatory. The Quartus II compiler errors out if this port is not instantiated and connected as shown in Figure 1.

The following example illustrates required modifications to the verilog wrapper file generated by the ALTLVDS MegaWizard Plug-In Manager configured in DPA mode with the **Use External PLL** option selected:

Example of modified verilog wrapper file (required modifications in green text)

```
module receiver_io (
        rx_enable,
        rx_in,
        rx_inclock,
        rx_reset,
        rx_dpa_locked,
        rx_out,
        rx_syncclock, // parallel clock input from ALTPLL
        pll_areset); // inverse of locked signal from ALTPLL

        input          rx_enable;
        input    [0:0]  rx_in;
        input          rx_inclock;
        input    [0:0]  rx_reset;
        input [0:0] pll_areset; //  declaration of pll_areset port
        input [0:0] rx_syncclock; // declaration of rx_syncclock port
        output   [0:0]  rx_dpa_locked;
        output   [3:0]  rx_out;

        wire [3:0] sub_wire0;
        wire [0:0] sub_wire1;
        wire [3:0] rx_out = sub_wire0[3:0];
        wire [0:0] rx_dpa_locked = sub_wire1[0:0];

        altlvds_rx  altlvds_rx_component (
                                .rx_inclock (rx_inclock),
                                .rx_reset (rx_reset),
                                .rx_in (rx_in),
                                .rx_enable (rx_enable),
                                .rx_out (sub_wire0),
                                .rx_dpa_locked (sub_wire1),
                                .pll_areset (pll_areset), // instantiation of pll_areset port
                                .rx_cda_max (),
                                .rx_cda_reset (1'b0),
                                .rx_channel_data_align (1'b0),
                                .rx_coreclk (1'b1),
                                .rx_data_align (1'b0),
                                .rx_data_align_reset (1'b0),
                                .rx_deskew (1'b0),
                                .rx_divfwdclk (),
                                .rx_dpll_enable (1'b1),
                                .rx_dpll_hold (1'b0),
                                .rx_dpll_reset (1'b0),
                                .rx_fifo_reset (1'b0),
                                .rx_locked (),
                                .rx_outclock (),
                                .rx_pll_enable (1'b1),
                                .rx_readclock (1'b0),
                                .rx_syncclock (rx_syncclock)); // instantiation of rx_syncclock  port
```

The following example illustrates the required ALTPLL-to-ALTLVDS connections in designs with ALTLVDS configured in DPA mode with the **Use External PLL** option selected:

Example of ALTPLL-to-ALTLVDS connections (required connections in green text)

```
module top(
            top_pll_areset,
            top_rx_refclk,
            top_rx_reset,
            top_rx_in
            );

input       top_pll_areset;
input       top_rx_refclk;
input       top_rx_reset;
input       top_rx_in;

wire serial_clk_pll_to_lvds;
wire load_enable_pll_to_lvds;
wire parallel_clk_pll_to_lvds;
wire rx_pll_locked_pll_to_lvds;

//ALTPLL receiver PLL  instantiation
pll_rx   pll_rx_inst
          (.areset(top_pll_areset),
          .inclk0(top_rx_refclk),
          .c0(serial_clk_pll_to_lvds),
          .c1(load_enable_pll_to_lvds),
          .c2(parallel_clk_pll_to_lvds),
          .locked(rx_pll_locked_pll_to_lvds)
          );

//ALTLVDS receiver channel  instantiation
receiver_io   receiver_io_inst
         (.rx_enable(load_enable_pll_to_lvds),
         .rx_inclock(serial_clk_pll_to_lvds),
         .rx_in(top_rx_in),
         .rx_reset(top_rx_reset),
         .rx_syncclock(parallel_clk_pll_to_lvds),
         .rx_out(lvds_rx_out),
         .pll_areset(~(rx_pll_locked_pll_to_lvds))
         );

endmodule
```

101 Innovation Drive
San Jose, CA 95134
www.altera.com