Determining Clock Skew

When the Virtex DLL Drives Multiple Copies of a Clock Off Chip

by Craig Abramson, Field Applications Engineer, craig.abramson@xilinx.com

High-speed logic boards often require multiple low-skew clock buffers to distribute high-speed clocks. With a little attention to detail, Virtex FPGAs can distribute the high-speed clocks for you.

With Virtex FPGAs, you can synchronize an external clock (driven out to your board) with an on-chip clock. Multiple on-chip DLL's make this possible. However, you may need several copies of the clock driven off-board, and using external clock buffer/drivers may negate any benefit of on-chip synchronization. You need a way to determine skew for multiple copies of a clock signal driven off-board by several IOBs.

Single Clock Driven Off-chip

Below is a block diagram of the architecture that would synchronize a clock driven off-chip to an on-board clock.

Single clock, CLK_P, being driven off chip.

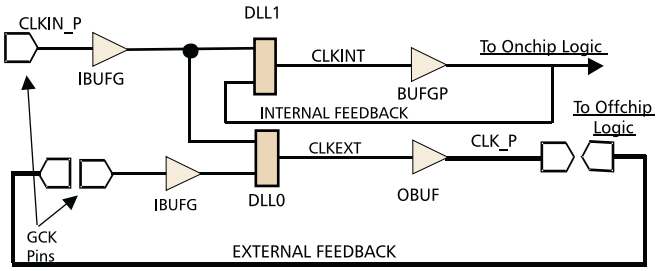

Figure 1

In the past, FPGA's were used primarily to consolidate all of a boards digital logic functions into a single device. With the new architectural features of the Virtex family, even more functionality can be pulled into the FPGA.

Multiple Clocks Driven Offchip

Let's look at an example where we want to replicate the output of DLL0, as shown in Figure 2.

Multiple copies of CLKEXT being driven offchip

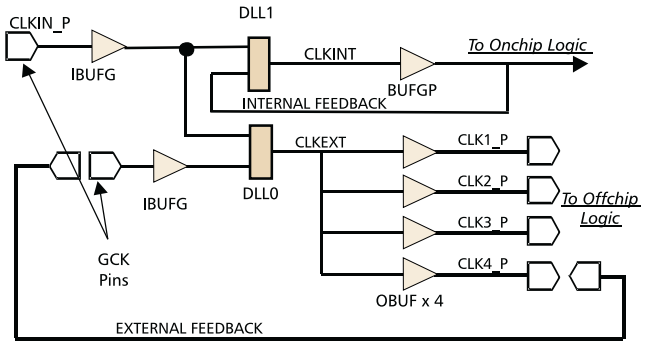

Figure 2

For the purpose of circuit analysis, you may want to minimize the skew on CLKEXT by specifying a constraint. Using the following constraint tells the place and route tools the amount of skew we can tolerate in the signal CLKEXT, which is the input to four different OBUFs.

```
NET CLKEXT MAXSKEW = 100ps ;
```

Note that the skew will be due to the differences in the routing distance from the source DLL's output to the inputs of the OBUFs. The delay through the OBUFs will contribute little to the total skew.

You also need to specify an intelligent choice of pin locations for CLK1_P, CLK2_P, CLK3_P, and CLK4_P. Selecting the four pins closest to DLL0's CLKIN pin gives the best results. (Viewing the chip in Epic can quickly tell you which bonded out IOBs are closest to the DLL). For a V300-BG352 device the UCF entries would look like this:

Continued on the following page

```
NET CLK1_P LOC = B15;
NET CLK2_P LOC = A15;
NET CLK3_P LOC = C13;
NET CLK4_P LOC = B13;
```

Running the design through the tools with the architecture shown in Figure 2, and using the listed location and skew constraints, yields the following results:

```
================================================================
Timing constraint: NET "CLKEXT" MAXSKEW = 100 pS ;
 1 item analyzed, 0 timing errors detected.
 Maximum net skew is 0.037ns.
--------------------------------------------------------------
Slack: 0.063ns CLKEXT
Report: 0.037ns skew meets 0.100ns timing con-
 straint by 0.063ns
From To Delay(ns) Skew(ns)
DLL3.CLK0 B15.O 1.153 0.000
DLL3.CLK0 C13.O 1.190 0.037
DLL3.CLK0 B13.O 1.153 0.000
DLL3.CLK0 A15.O 1.190 0.037
```

Be sure to select "Report Paths in Timing Constraints" in the Timing Reports Tab to direct the Xilinx tools to report everything that's covered by timing constraints (such as the skew of CLKEXT in this example).

Minimizing skew internal to the FPGA is only part of the entire system level skew problem. During board layout you must carefully match the external clock net delays as well. Also, be sure to bring the DLL's clock and external feedback signal in through GCK pins and IBUFG pins as illustrated.

Examine the Design in Epic

By examining the design in Epic, you can see that the careful selection of IOBs made it very easy for the place and route tools to meet the skew specification. If you select the net CLKEXT in the "Epic List" window, you can see that the routes connecting the DLL's output to the input of each of the four IOBs appear to be fairly equal in length. This will translate into a very low skew. To confirm the numbers reported above, while still in Epic and with the CLKEXT net still selected, press the "DELAY" button. The delays associated with the CLKEXT will scroll by in the Epic window.

EPIC view of DLL and associated IOBs.

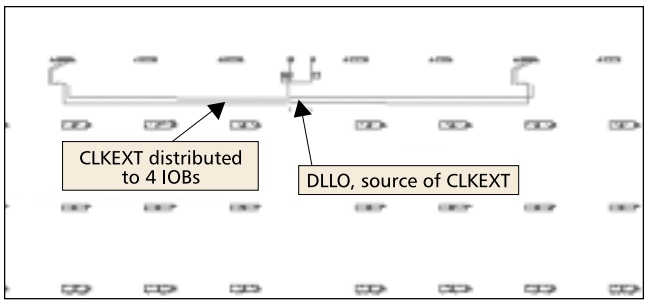

Figure 3

If more copies of the clock need to be driven off-chip, you should first examine the chip in EPIC to determine which I/O locations should be selected. Then, by simply adding the location constraints to your User Constraints File, and a few lines of code to your VHDL or Verilog file, you can perform the same skew analysis.

Code Example

The following code demonstrates the instantiations necessary to drive four IOBs with CLKEXT.

Verilog code example of DLL driving for IOBs.

```
// ----------------------------
// Simple desgn that allows analysis of skew of
// DLL output being distributed to several IOBs.
// ----------------------------
module skewtest (CLKIN_P,CKL1_P,CLK2_P,CLK4_P,cklfbin_p
input CLKIN_P; //Drive the DLLs
input clkfbin_p; //Pin in for external feedback
output CLK1_P;
output CLK2_P;
output CLK3_P;
output CLK4_P;
wire CLKIN,CLKINT,CLKEXT,CLK,clkfbin;

// ----------------------------
// The ibufg is needed to drive the clock input to both DLLs
// as well as the feedback coming from clocks driven off chip.
// ----------------------------
IBUFG bugfio0 (.I(CLKIN_P),.O(CLKIN));
IBUFG buggio1 (.I(clkfbin_p),.O(clkfbin));

// ----------------------------
// DLL0 drives external clock
// ----------------------------
CLKDLL dll0(
  .CLKIN(CLKIN),.CLKFB(clkfbin),.RST(logic0),
  .CLK0(CLKEXT),.slowclk(),.CLK180(),.CLK270(),
  .CLK2X(),.CLKDV(),.LOCKED()
  );

// ----------------------------
// DLL1 drives internal logic
// ----------------------------
CLKDLL dll1(
  .CLKIN(CLKIN),.CLKFB(clkfbin),.RST(logic0),
  .CLK0(CLKINT),.slowclk(),.CLK180(),.CLK270(),
  .CLK2X(),.CLKDV(),.LOCKED()
  );

// ----------------------------
// OBUF (output buffer) is needed to drive a synchronized clock off chip
// ----------------------------
OBUF obuf0 (.I(CLKEXT),.O(CLK1_P));
OBUF obuf1 (.I(CLKEXT),.O(CLK2_P));
OBUF obuf2 (.I(CLKEXT),.O(CLK3_P));
OBUF obuf3 (.I(CLKEXT),.O(CLK4_P));
endmodule
```

Figure 4

Conclusion

In the past, FPGA's were used primarily to consolidate all of a boards digital logic functions into a single device. With the new architectural features of the Virtex family, even more functionality can be pulled into the FPGA. In this example, distributing the DLL output among four carefully chosen IOBs yields 37 ps of skew. This is comparable to the low skew clock drivers currently on the market. In addition the clock signals being driven off chip can be 1 of 13 different I/O voltage standards. So in this scenario, two functions that would ordinarily be handled by devices external to the FPGA are now performed by the Virtex device.

For more information on creating low-skew clocks using the Virtex DLL's, see www.xilinx.com/xapp/xapp132.pdf. For more information on the many I/O standards supported by Virtex FPGAs, see www.xilinx.com/xapp/xapp133.pdf Σ