XAPP174 (v1.1) January 24, 2000

Using Delay-Locked Loops in Spartan-II FPGAs

Summary

The Spartan™-II family provides four fully digital dedicated on-chip Delay-Locked Loop (DLL) circuits, which provide zero propagation delay, low clock skew between output clock signals distributed throughout the device, and advanced clock domain control. These dedicated DLLs can be used to implement several circuits that improve and simplify system level design.

Introduction

Supporting the highest bandwidth data rates between devices requires advanced clock management technology. Clock delay and clock skew are key contributors to device performance. Delay-Locked Loop (DLL) circuits in the Spartan-II FPGAs provide zero propagation delay and low clock skew between output clock signals distributed throughout the device. Each Spartan-II device has four fully digital dedicated on-chip DLLs, allowing for very precise synchronization of external and internal clocks.

Each DLL can drive up to two global clock routing networks within the device. The global clock distribution network minimizes clock skews due to loading differences. By monitoring a sample of the DLL output clock, the DLL can compensate for the delay on the routing network, effectively eliminating the delay from the external input port to the individual clock loads within the device.

In addition to providing zero delay with respect to a user source clock, the DLL can provide multiple phases of the source clock. The DLL can also act as a clock doubler or it can divide the user source clock by up to 16.

Clock multiplication gives the designer a number of design alternatives. For instance, a 50 MHz source clock doubled by the DLL can drive an FPGA design operating at 100 MHz. This technique can simplify board design because the clock path on the board no longer distributes such a high-speed signal. A multiplied clock also provides designers the option of time-domain-multiplexing, using one circuit twice per clock cycle, consuming less area than two copies of the same circuit.

The DLL can also act as a clock mirror. By driving the DLL output off-chip and then back in again, the DLL can be used to deskew a board level clock between multiple devices.

In order to guarantee the system clock establishes prior to the device "waking up," the DLL can delay the completion of the device configuration process until after the DLL achieves lock.

By taking advantage of the DLL to remove on-chip clock delay, the designer can greatly simplify and improve system level design involving high-fanout, high-performance clocks.

Fundamentals

Two basic types of circuits remove clock delay: Phased-Locked Loops (PLLs), and DLLs. In addition to the primary function of removing clock distribution delay, DLLs and PLLs typically provide some additional functionality such as frequency synthesis (clock multiplication and clock division) and clock conditioning (duty cycle correction and phase shifting). Multiple clock outputs can also be deskewed with respect to one another, to take advantage of multiple clock domains.

Delay-Locked Loop

As shown in Figure 1, a DLL in its simplest form consists of a variable delay line and control logic. The delay line produces a delayed version of the input clock CLKIN. The clock distribution network routes the clock to all internal registers and to the clock feedback CLKFB pin. The control logic must sample the input clock as well as the feedback clock in order to adjust the delay line.

Delay lines can be built using a voltage controlled delay or a series of discrete delay elements. For optimum performance the Xilinx DLL uses a discrete digital delay line.

A DLL works by inserting delay between the input clock and the feedback clock until the two rising edges align, putting the two clocks 360 degrees out of phase (meaning they are in phase). After the edges from the input clock line up with the edges from the feedback clock, the DLL "locks." As long as the circuit is not evaluated until after the DLL locks, the two clocks have no discernible difference. Thus, the DLL output clock compensates for the delay in the clock distribution network, effectively removing the delay between the source clock and its loads.

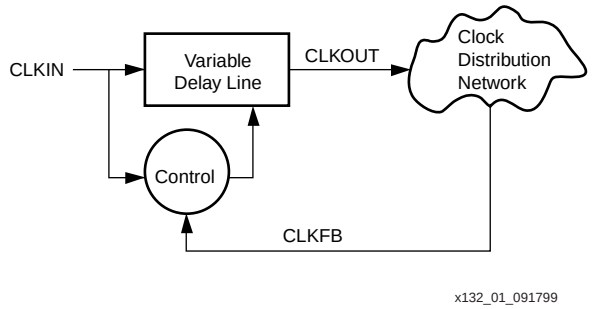

Figure 1: **Delay-Locked Loop Block Diagram**

Phase-Locked Loop

While designed for the same basic function, the PLL uses a different architecture to accomplish the task. As shown in Figure 2, the fundamental difference between the PLL and DLL is that instead of a delay line, the PLL uses a voltage controlled oscillator, which generates a clock signal that approximates the input clock CLKIN. The control logic, consisting of a phase detector and filter, adjusts the oscillator frequency and phase to compensate for the clock distribution delay.

The PLL control logic compares the input clock to the feedback clock CLKFB and adjusts the oscillator clock until the rising edge of the input clock aligns with the feedback clock. The PLL then "locks."

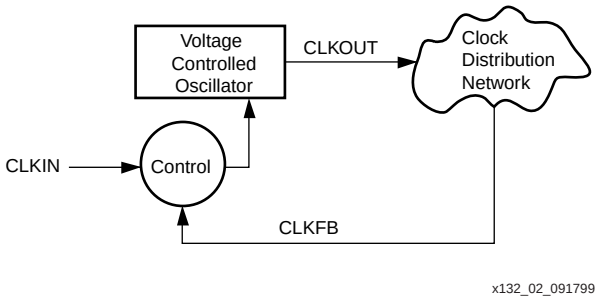

Figure 2: **Phase-Locked Loop Block Diagram**

Implementation

Implementation of the DLL or PLL can be accomplished using either analog or digital circuitry; each holds its own advantages. An analog implementation with careful design can produce a DLL or PLL with a finer timing resolution. Analog implementations can additionally take less silicon area.

Conversely, digital implementations offer advantages in noise sensitivity, lower power consumption and superior jitter performance. Digital implementations also provide the ability to stop the clock, facilitating power management. Analog implementations can require additional power supplies, require close control of the power supply, and pose problems in migration to new process technologies.

DLL vs. PLL

When it comes to choosing between a PLL or a DLL for a particular application, understand the differences in the architectures. The oscillator used in the PLL inherently introduces instability and an accumulation of phase error. This in turn degrades the performance of the PLL when attempting to compensate for the delay of the clock distribution network. Conversely, the unconditionally stable DLL architecture does not accumulate phase error.

For this reason, for delay compensation and clock conditioning, choose the DLL as the architecture.

Library DLL Symbols

Figure 3 shows the simplified Xilinx library DLL macro symbol, BUFGDLL. This macro delivers a quick and efficient way to provide a system clock with zero propagation delay throughout the device. Figure 4 and Figure 5 show the two library DLL primitives. These symbols provide access to the complete set of DLL features when implementing more complex applications.

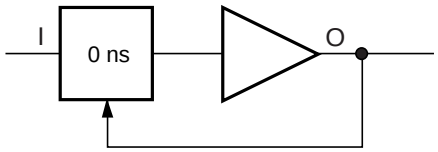

Figure 3: **Simplified DLL Macro Symbol BUFGDLL**

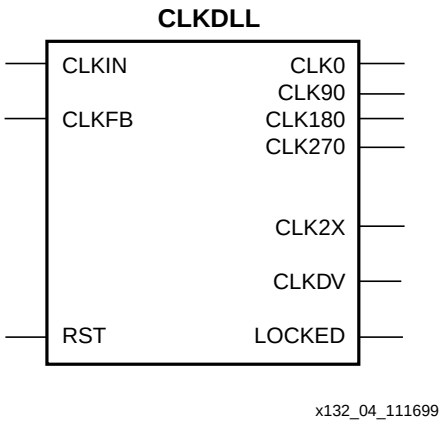

Figure 4: **Standard DLL Symbol CLKDLL**

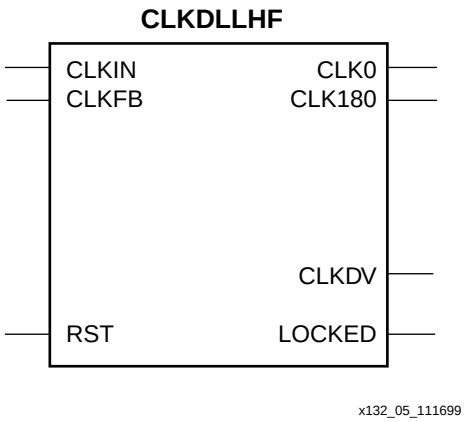

Figure 5: **High-frequency DLL Symbol CLKDLLHF**

BUFGDLL Macro Pin Descriptions

Use the BUFGDLL macro as the simplest way to provide zero propagation delay for a high-fanout on-chip clock from an external input. This macro uses the IBUFG, CLKDLL and BUFG primitives to implement the most basic DLL application as shown in Figure 6.

The IBUFG is a dedicated clock input pin for a DLL. There are four of these pins, one for each DLL. They are input only and can use one of the many input voltage standards available. Banking rules will need to be followed and the associated bank number can be found in the datasheet. The IBUFG has a dedicated route to its associated DLL. This ensures smallest delay and increased accuracy in use of the DLL. However, the DLL inputs can be driven by other device pins.

BUFG is the global clock buffer, which in turn drives high-speed, low-skew signals throughout the device. The IBUFG, CLKDLL, and BUFG must all be on the same edge of the device (top or bottom).

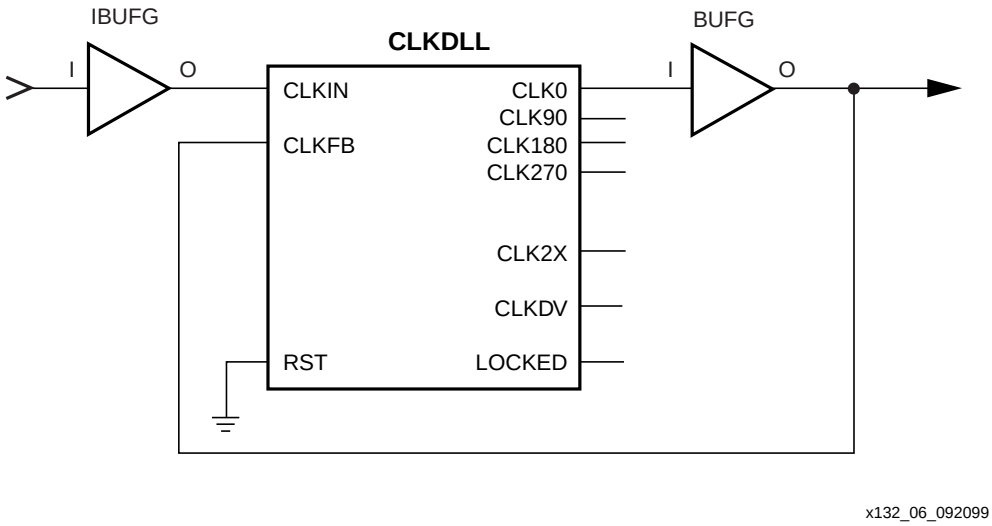

Figure 6: **BUFGDLL Schematic**

This symbol does not provide access to the advanced clock domain controls or to the clock multiplication or clock division features of the DLL. This symbol also does not provide access to the RST or LOCKED pins of the DLL. For access to these features, a designer must use the library DLL primitives described in the following sections.

Source Clock Input — I

The I pin provides the user source clock, the clock signal on which the DLL operates, to the BUFGDLL. For the BUFGDLL macro the source clock frequency must fall in the low frequency range as specified in the datasheet. The BUFGDLL requires an external signal source clock. Therefore, only an external input port can source the signal that drives the BUFGDLL I pin.

Clock Output — O

The clock output pin O represents a delay-compensated version of the source clock (I) signal. This signal, sourced by a global clock buffer BUFG symbol, takes advantage of the dedicated global clock routing resources of the device.

The output clock has a 50/50 duty cycle unless you deactivate the duty cycle correction property.

CLKDLL Primitive Pin Descriptions

The library CLKDLL primitives provide access to the complete set of DLL features needed when implementing more complex applications with the DLL.

Source Clock Input — CLKIN

The CLKIN pin provides the user source clock (the clock signal on which the DLL operates) to the DLL. The CLKIN frequency must fall in the ranges specified in the datasheet. Either a global clock buffer (BUFG) driven from another CLKDLL or one of the global clock input buffers (IBUFG) on the same edge of the device (top or bottom) must source this clock signal.

Feedback Clock Input — CLKFB

The DLL requires a reference or feedback signal to provide the delay-compensated output. Connect only the CLK0 or CLK2X DLL outputs to the feedback clock input (CLKFB) pin to provide the necessary feedback to the DLL. Either a global clock buffer (BUFG) or one of the global clock input buffers (IBUFG) on the same edge of the device (top or bottom) must source this clock signal.

If an IBUFG sources the CLKFB pin, the following special rules apply.

1. An external input port must source the signal that drives the IBUFG I pin.
2. The CLK2X output must feed back to the device if both the CLK0 and CLK2X outputs are driving off chip devices.
3. That signal must directly drive only OBUFs and nothing else.

These rules enable the software to determine which DLL clock output sources the CLKFB pin.

Reset Input — RST

When the reset pin RST activates, the LOCKED signal deactivates within four source clock cycles. The RST pin, active High, must either connect to a dynamic signal or be tied to ground. As the DLL delay taps reset to zero, glitches can occur on the DLL clock output pins. Activation of the RST pin can also severely affect the duty cycle of the clock output pins. Furthermore, the DLL output clocks no longer deskew with respect to one another. For these reasons, rarely use the reset pin unless reconfiguring the device or changing the input frequency.

2x Clock Output — CLK2X

The output pin CLK2X provides a frequency-doubled clock with an automatic 50/50 duty-cycle correction. Until the CLKDLL has achieved lock, the CLK2X output appears as a 1x version of the input clock with a 25/75 duty cycle. This behavior allows the DLL to lock on the correct edge with respect to source clock. This pin is not available on the CLKDLLHF primitive.

Clock Divide Output — CLKDV

The clock divide output pin CLKDV provides a lower frequency version of the source clock. The CLKDV_DIVIDE property controls CLKDV such that the source clock is divided by N where N is either 1.5, 2, 2.5, 3, 4, 5, 8, or 16.

This feature provides automatic duty cycle correction such that the CLKDV output pin always has a 50/50 duty cycle.

1x Clock Outputs — CLK[0|90|180|270]

The 1x clock output pin CLK0 represents a delay-compensated version of the source clock (CLKIN) signal. The CLKDLL primitive provides three phase-shifted versions of the CLK0 signal while CLKDLLHF provides only the 180 degree phase-shifted version. The relationship between phase shift and the corresponding period shift appears in Table 1.

Table 1: **Relationship of Phase-shifted Output Clock to Period Shift**

Phase (degrees)	% Period Shift
0	0%
90	25%
180	50%
270	75%

The timing diagrams in Figure 7 illustrate the DLL clock output characteristics.

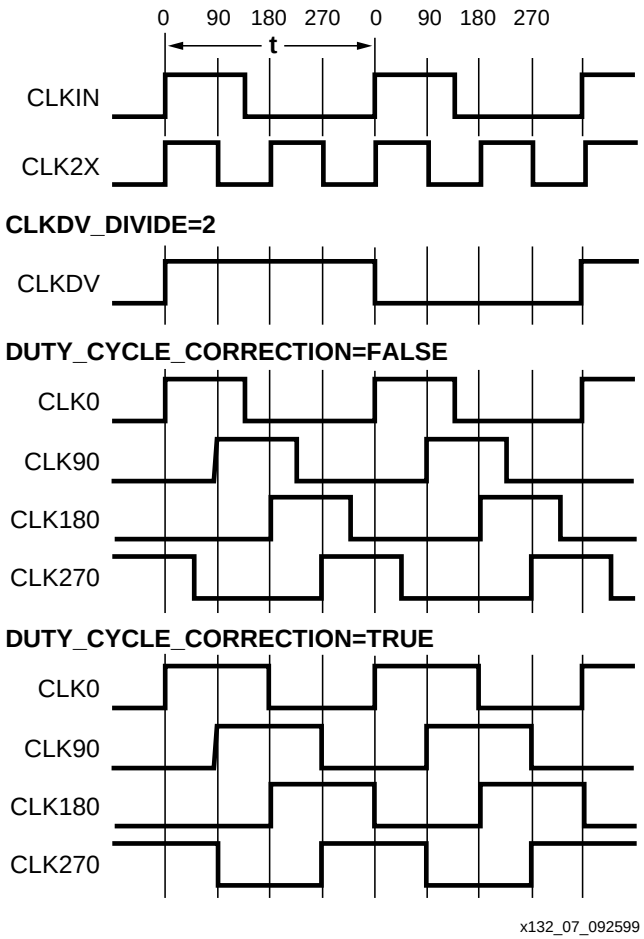

Figure 7: **DLL Output Characteristics**

The DLL provides duty cycle correction on all 1x clock outputs such that all 1x clock outputs by default have a 50/50 duty cycle. The DUTY_CYCLE_CORRECTION property (TRUE by default) controls this feature. In order to deactivate the DLL duty cycle correction, attach the DUTY_CYCLE_CORRECTION=FALSE property to the DLL symbol. When duty cycle correction deactivates, the output clock has the same duty cycle as the source clock.

The DLL clock outputs can drive an OBUF, a BUFG, or they can route directly to destination clock pins. The DLL clock outputs can only drive the BUFGs that reside on the same edge (top or bottom).

Locked Output — LOCKED

In order to achieve lock, the DLL may need to sample several thousand clock cycles. After the DLL achieves lock the LOCKED signal activates. The DLL timing parameter section of the datasheet provides estimates for locking times.

In order to guarantee that the system clock is established prior to the device "waking up," the DLL can delay the completion of the device configuration process until after the DLL locks. The STARTUP_WAIT property activates this feature.

Until the LOCKED signal activates, the DLL output clocks are not valid and can exhibit glitches, spikes, or other spurious movement. In particular the CLK2X output will appear as a 1x clock with a 25/75 duty cycle.

DLL Properties

Properties provide access to some of the Spartan-II Family DLL features, (for example, clock division and duty cycle correction).

Duty Cycle Correction Property

The 1x clock outputs, CLK0, CLK90, CLK180, and CLK270, use the duty cycle corrected default such that they exhibit a 50/50 duty cycle. The DUTY_CYCLE_CORRECTION property (by default TRUE) controls this feature. In order to deactivate the DLL duty cycle correction for the 1x clock outputs, attach the DUTY_CYCLE_CORRECTION=FALSE property to the DLL symbol. When duty cycle correction deactivates, the output clock has the same duty cycle as the source clock.

Clock Divide Property

The CLKDV_DIVIDE property specifies how the signal on the CLKDV pin is frequency divided with respect to the CLK0 pin. The values allowed for this property are 1.5, 2, 2.5, 3, 4, 5, 8, or 16; the default value is 2.

Startup Delay Property

This property, STARTUP_WAIT, takes on a value of TRUE or FALSE (the default value). When TRUE the device configuration DONE signal waits until the DLL locks before going to High.

Spartan-II Family DLL Location Constraints

The DLLs are distributed such that there is one DLL in each corner of the device. The location constraint LOC, attached to the DLL symbol with the numeric identifier 0, 1, 2, or 3, controls the DLL location. The orientation of the four DLLs and their corresponding clock resources appears in Figure 8.

The LOC property uses the following form: `LOC=DLL2`.

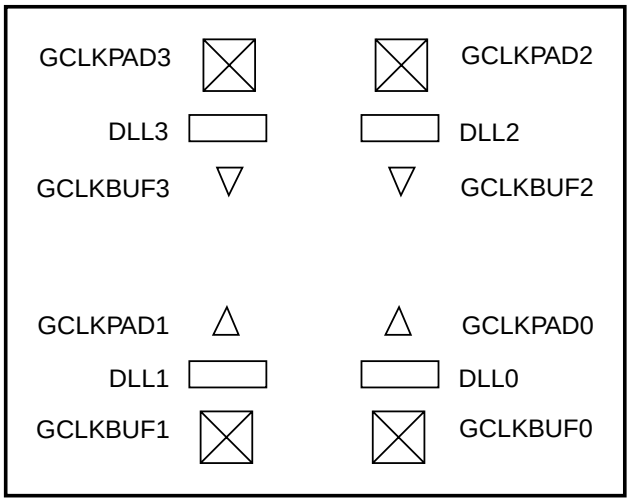

Figure 8: **Orientation of Spartan-II Family DLLs**

Design Considerations

Use the following design considerations to avoid pitfalls and improve success designing with Xilinx devices.

Input Clock

The output clock signal of a DLL, essentially a delayed version of the input clock signal, reflects any instability on the input clock in the output waveform. For this reason the quality of the DLL input clock relates directly to the quality of the output clock waveforms generated by the DLL. The DLL input clock requirements are specified in the datasheet.

In most systems a crystal oscillator generates the system clock. The DLL can be used with any commercially available quartz crystal oscillator. For example, most crystal oscillators produce an output waveform with a frequency tolerance of 100 PPM, meaning 0.01 percent change in the clock period. The DLL operates reliably on an input waveform with a frequency drift of up to 1 ns — orders of magnitude in excess of that needed to support any crystal oscillator in the industry. However, the cycle-to-cycle jitter must be kept to less than 300 ps in the low frequencies and 150 ps for the high frequencies.

Input Clock Changes

Changing the period of the input clock beyond the maximum drift amount requires a manual reset of the CLKDLL. Failure to reset the DLL will produce an unreliable lock signal and output clock.

It is possible to stop the input clock with little impact to the DLL. Stopping the clock should be limited to less than 100 µs to keep device cooling to a minimum. The clock should be stopped during a Low phase, and when restored the full High period should be seen. During this time LOCKED will stay High and remain High when the clock is restored.

When the clock is stopped, one to four more clocks will still be observed as the delay line is flushed. When the clock is restarted, the output clocks will not be observed for one to four clocks as the delay line is filled. The most common case will be two or three clocks.

In a similar manner, a phase shift of the input clock is also possible. The phase shift will propagate to the output one to four clocks after the original shift, with no disruption to the CLKDLL control.

Output Clocks

As mentioned earlier in the DLL pin descriptions, some restrictions apply regarding the connectivity of the output pins. The DLL clock outputs can drive an OBUF, a global clock buffer BUFG, or route directly to destination clock pins. The only BUFGs that the DLL clock outputs can drive are the two on the same edge of the device (top or bottom).

Do not use the DLL output clock signals until after activation of the LOCKED signal. Prior to the activation of the LOCKED signal, the DLL output clocks are not valid and can exhibit glitches, spikes, or other spurious movement.

Useful Application Examples

The DLL can be used in a variety of creative and useful applications. The following examples show some of the more common applications. The Verilog and VHDL example files are available at: http://www.xilinx.com/bvdocs/appnotes/xapp174.zip

Standard Usage

The circuit shown in Figure 9 resembles the BUFGDLL macro implemented in such a way as to provide access to the RST and LOCKED pins of the CLKDLL.

The dll_standard files in the `xapp174.zip` file show the VHDL and Verilog implementation of this circuit.

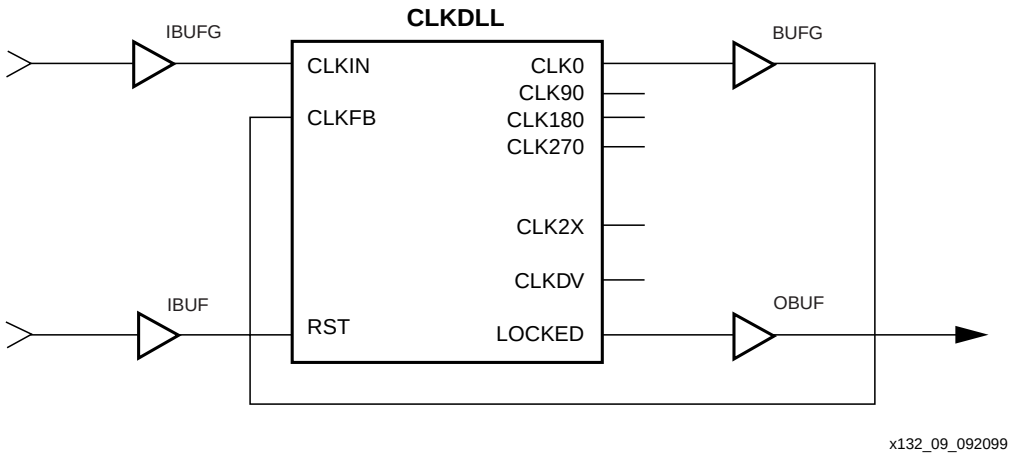

Figure 9: **Standard DLL Implementation**

Board Level Deskew of Multiple Non-Spartan-II Devices

The circuit shown in Figure 10 can be used to deskew a system clock between a Spartan-II chip and other non-Spartan-II chips on the same board. This application is commonly used when the Spartan-II device is used in conjunction with other standard products such as SRAM or DRAM devices. While designing the board level route, ensure that the return net delay to the source equals the delay to the other chips involved.

Do not use the DLL output clock signals until after activation of the LOCKED signal. Prior to the activation of the LOCKED signal, the DLL output clocks are not valid and can exhibit glitches, spikes, or other spurious movement.

The `dll_mirror_1` files in the `xapp174.zip` file show the VHDL and Verilog implementation of this circuit.

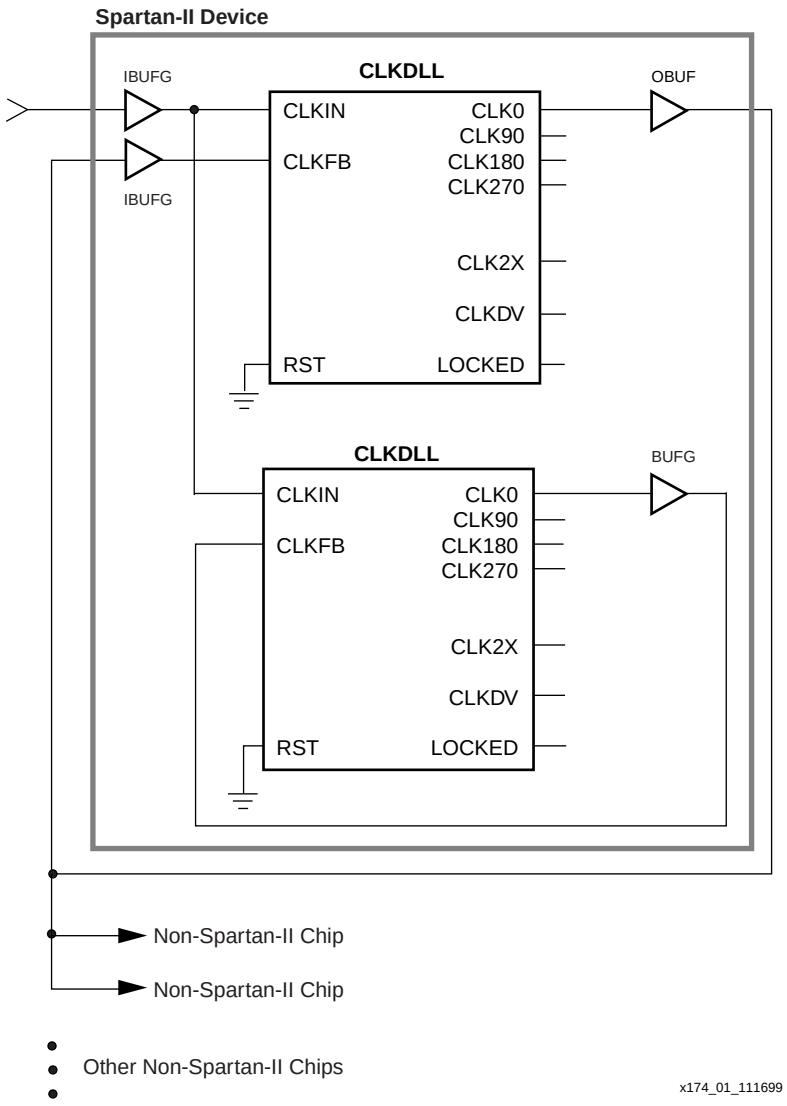

Figure 10: **DLL Deskew of Board Level Clock Between Spartan-II and Multiple Non-Spartan-II Devices**

Board Level Deskew of Multiple Spartan-II Devices

The circuit shown in Figure 11 can be used to deskew a system clock between multiple Spartan-II chips on the same board. While designing the board level route, ensure that the return net delay to the source equals the delay to the other chips involved.

Do not use the DLL output clock signals until after activation of the LOCKED signal. Prior to the activation of the LOCKED signal, the DLL output clocks are not valid and can exhibit glitches, spikes, or other spurious movement.

The `dll_mirror_2` files in the `xapp174.zip` file show the VHDL and Verilog implementation of this circuit.

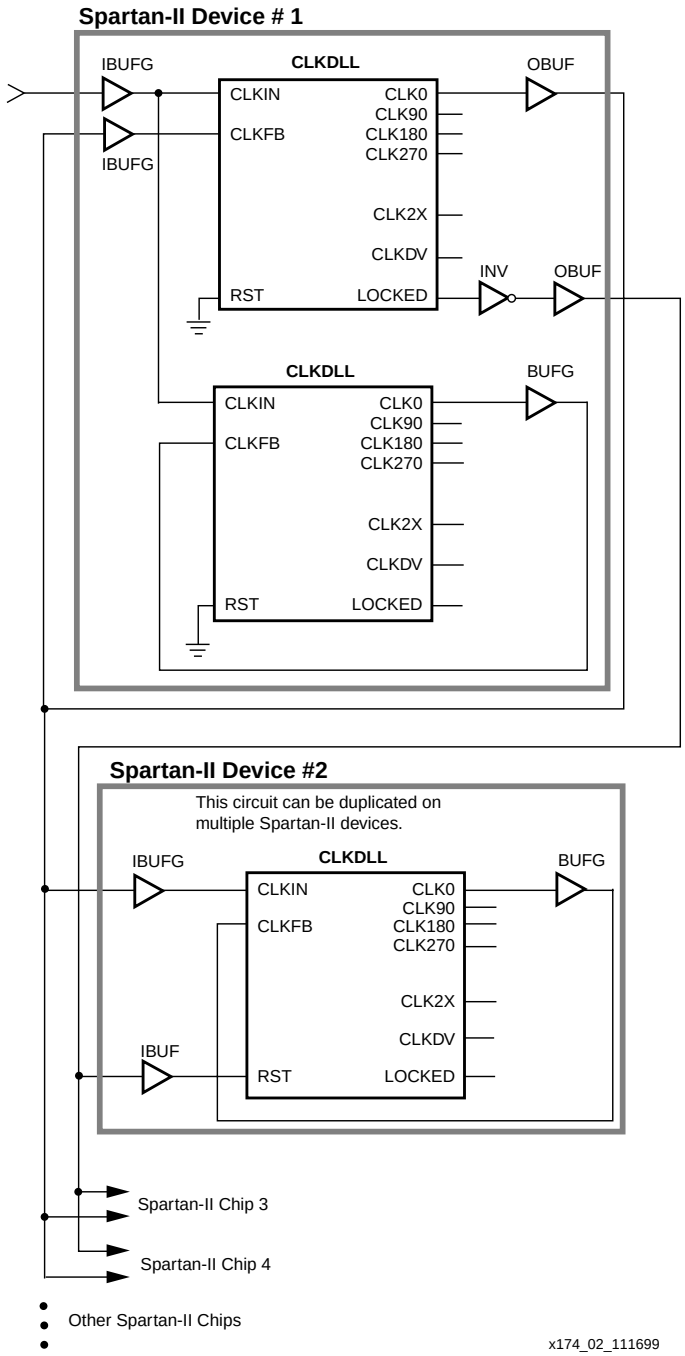

Figure 11: **DLL Deskew of Board Level Clock Between Multiple Spartan-II Devices**

Deskew of Clock and Its 2x Multiple

The circuit shown in Figure 12 implements a 2x clock multiplier and also uses the CLK0 clock output with zero skew between registers on the same chip. A clock divider circuit could alternatively be implemented using similar connections.

Because any single DLL can only access at most two BUFGs, any additional output clock signals must be routed from the DLL in this example on the high speed backbone routing.

The `dll_2x` files in the `xapp174.zip` file show the VHDL and Verilog implementation of this circuit.

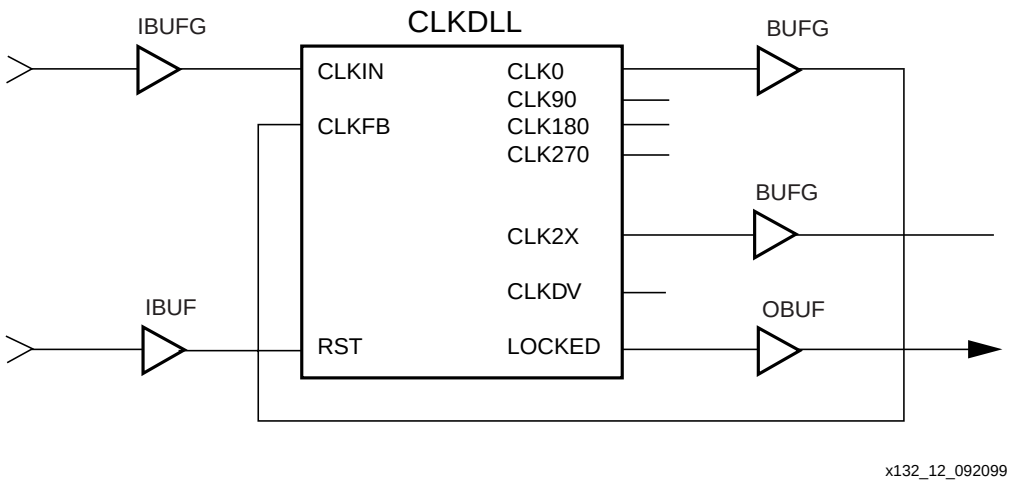

Figure 12: **DLL Deskew of Clock and 2x Multiple**

Generating a 4x Clock

By connecting two DLL circuits each implementing a 2x clock multiplier in series as shown in Figure 13, a 4x clock multiply can be implemented with zero skew between registers in the same device.

If another clock output is needed, the clock could access a BUFG only if the DLLs are constrained to exist on opposite edges (top or bottom) of the device.

When using this circuit it is vital to use the SRL16 cell to reset the second DLL after the initial chip reset. If this is not done, the second DLL may not recognize the change of frequencies when the input changes from a 1x (25/75) waveform to a 2x (50/50) waveform.

The `dll_4x` files in the `xapp174.zip` file show the VHDL and Verilog implementation of this circuit.

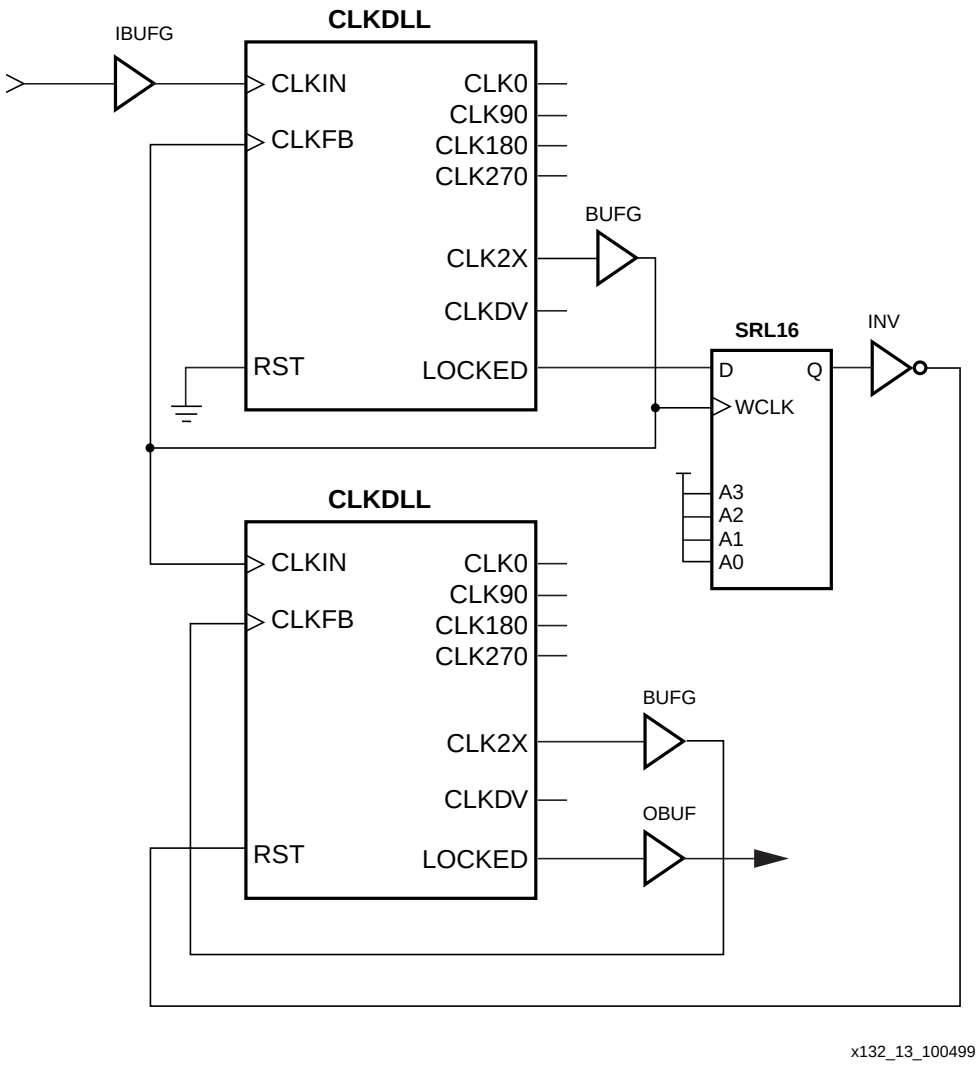

Figure 13: **DLL Generation of 4x Clock**

Revision History

The following table shows the revision history for this document.

Date	Version #	Revision
11.23.99	1.0	Initial Xilinx release.
01.24.00	1.1	Added generation of a 4x clock.