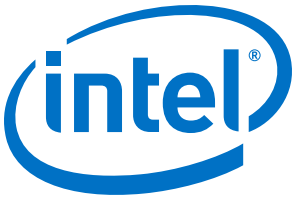


# Intel® Agilex™ Clocking and PLL User Guide

Updated for Intel® Quartus® Prime Design Suite: **19.3**



UG-20216 | 2019.12.18
Latest document on the web: PDF | HTML

# Contents

# 1. Intel® Agilex™ Clocking and PLL Overview

## 1.1. Clock Networks Overview

Intel® Agilex™ devices contain dedicated resources for distributing signals throughout the fabric. These resources are typically used for clock signals and other signals with low-skew requirements. In Intel Agilex devices, these resources are implemented as a programmable clock routing network, which allows for the implementation of various low-skew clock trees.

**Related Information**

Use Global Clock Network Resources, Design Recommendations User Guide (Intel Quartus® Prime Pro Edition)
Provides more information about clock assignments in the Intel Quartus® Prime software.

## 1.2. PLLs Overview

Phase-locked loops (PLLs) provide robust clock management and synthesis for device clock management, external system clock management, and high-speed I/O interfaces.

The Intel Agilex device family contains the following I/O PLLs for core applications. The I/O PLLs can only function as integer PLLs.

- Fabric-feeding I/O PLLs—three C counter outputs available and do not support PLL cascading
- I/O bank I/O PLLs—seven C counter outputs available and support PLL cascading

The I/O PLLs are located adjacent to the hard memory controllers and LVDS serializer/deserializer (SERDES) blocks in the I/O banks. Each I/O bank contains two I/O bank I/O PLLs and one fabric-feeding I/O PLL.

# 2. Intel Agilex Clocking and PLL Architecture and Features

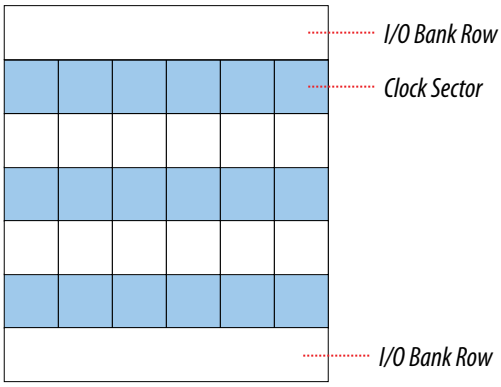
## 2.1. Clock Networks Architecture and Features

### 2.1.1. Clock Network Architecture

Each Intel Agilex device is divided into a number of evenly sized clock sectors.

**Figure 1. Clock Sector Floorplan for Intel Agilex Devices**

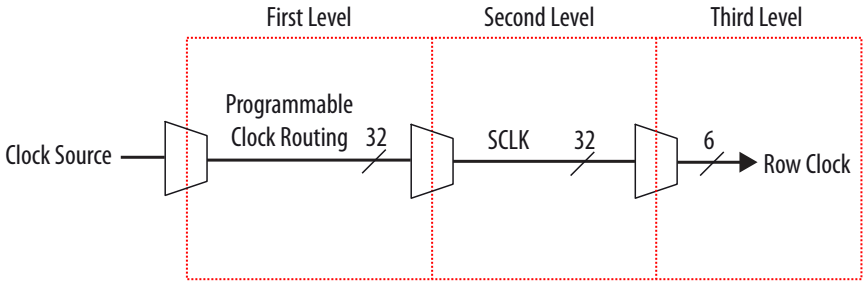
This figure shows an example of the clock sectors in an Intel Agilex device, which is implemented as an array of sectors—5 rows and 6 columns in this example. I/O banks are at the top and bottom of the Intel Agilex device.




#### 2.1.1.1. Clock Network Hierarchy

The Intel Agilex clock network is organized in a hierarchy with 3 levels.

**Figure 2. Clock Network Hierarchy**

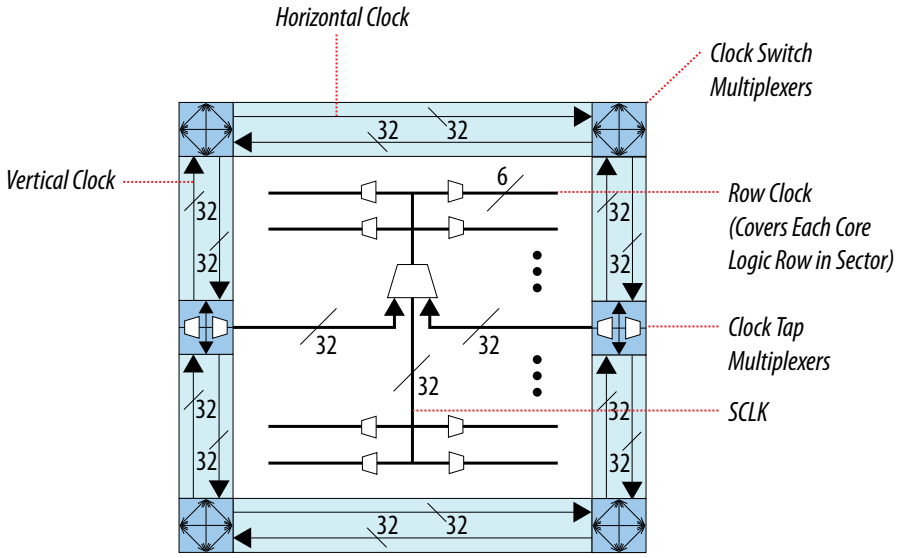


### 2.1.1.2. Clock Sector

Each clock sector has a dedicated sector clock (SCLK) network and a row clock network that can be accessed by the programmable clock routing. On each side of the clock sector, there is a channel that contains 64 unidirectional wires in bidirectional pairs, where only one wire in each pair can be used at one time. At each corner, there is a set of programmable clock switch multiplexers that can route between these clock wires.

A signal on a vertical clock wire can enter the sector to its left or right via clock tap multiplexers. The clock tap multiplexer drives a sector clock, which distributes the signal to each row in the clock sector. In each row, there are six row clock resources that route to all core functional blocks, PLLs, and I/O interfaces in the sector, and to adjacent transceivers.

**Figure 3. Dedicated Clock Resources Within a Clock Sector**




### 2.1.1.3. Programmable Clock Routing

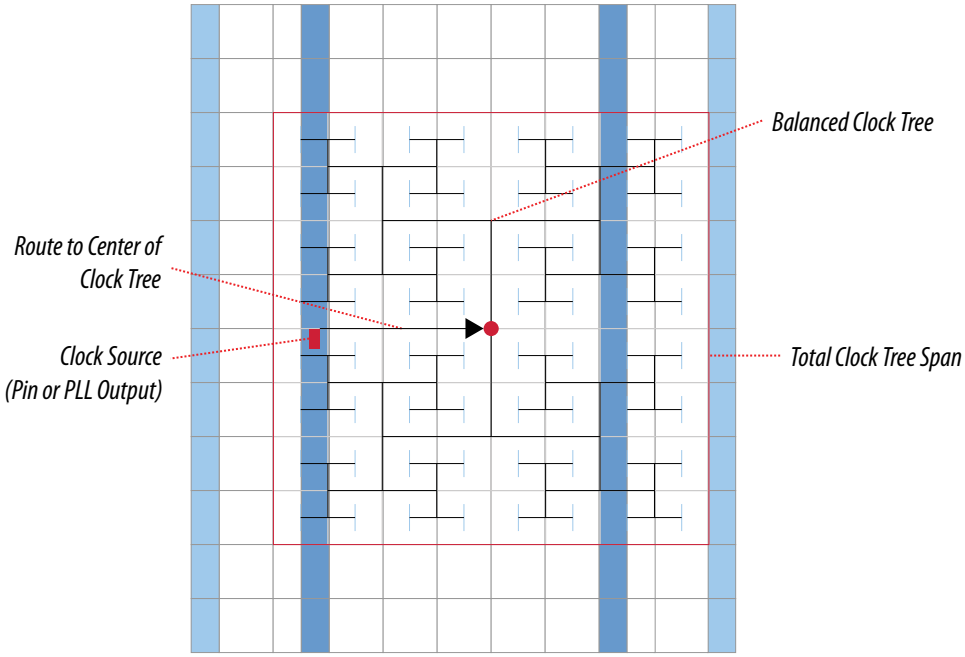
The Intel Quartus® Prime software automatically configures the clock switch multiplexer, clock tap multiplexer, SCLK multiplexer, and row clock multiplexers to generate skew-balanced clock trees. The resulting routing path distributes the signal from the clock source to all target destinations in one or more clock sectors.

The Intel Quartus Prime software creates efficiently balanced clock trees of various sizes, ranging from a single clock sector to the entire device, as shown in the following figure. By default, the Intel Quartus Prime software automatically determines the size and location of the clock tree. Alternatively, you can directly constrain the clock tree size and location by using either a Clock Region Assignment or Logic Lock Regions.

The total insertion delay for the clock network depends on the number of clock resources needed to implement the clock tree, increasing with the distance of the furthest clock destination from the signal source. As delay increases, the worst-case skew for crossing clock networks using different clock tree branches grows, potentially degrading the maximum performance. For very high-speed clock signals, it is

advantageous to reduce the number of clock networks driven, which reduces the clock skew, and to reduce the distance between the clock source and the furthest destination, which reduces both clock skew and total clock insertion delay.

**Figure 4. Examples of Clock Networks Sizes Using Intel Agilex Programmable Clock Routing**




## 2.1.2. Clock Resources

**Table 1. Programmable Clock Routing Resources for Intel Agilex Devices**

| Number of Resources Available | Source of Clock Resource |
|---|---|
| 32 pairs of unidirectional programmable clock routing at the boundary of each clock sector | For transceiver bank:<br>• Physical medium attachment (PMA) and physical coding sublayer (PCS) TX and RX clocks per channel<br>• PMA and PCS TX and RX divide clocks per channel<br>• Hard IP core clock output signals<br>• REFCLK pins<br>• Core signals (1)<br>For I/O bank:<br>• I/O PLL C counter outputs<br>• I/O PLL M counter outputs for feedback<br>• Phase aligner counter outputs<br>• Dynamic phase alignment (DPA) clock output<br>• Clock input pins<br>• Core signals (1) |

(1) Core signals drive directly to programmable clock routing through clock switch multiplexers in the clock sector instead of the periphery DCM block.

For more information about the clock input pins connections, refer to the pin connection guidelines.

**Related Information**

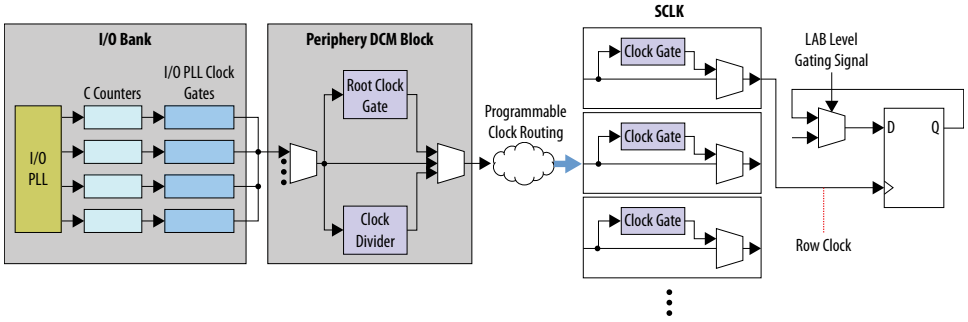
Intel Agilex Device Family Pin Connection Guidelines

## 2.1.3. Clock Control Features

The following figure shows the high level description of the Intel Agilex clock control features—clock gating and clock divider. The clock from the I/O PLL output can be gated dynamically. These clock signals along with other clock sources go to the periphery distributed clock multiplexer (DCM). In the periphery DCM, the clock signal can either pass straight through, be gated by the root clock gate, or be divided by the clock divider.

The Intel Quartus Prime software routes the clock signal on the programmable clock routing to reach each clock sector. The clock signal can be gated in each sector by the SCLK gates. The clock enters the SCLK network followed by the row clock network, and eventually reaches the registers in the core. The LAB registers have a built-in functional clock enable feature, as shown in the following figure.

**Figure 5. Clock Gating and Clock Divider in Intel Agilex Clock Network**




### 2.1.3.1. Clock Gating

#### 2.1.3.1.1. Root Clock Gate

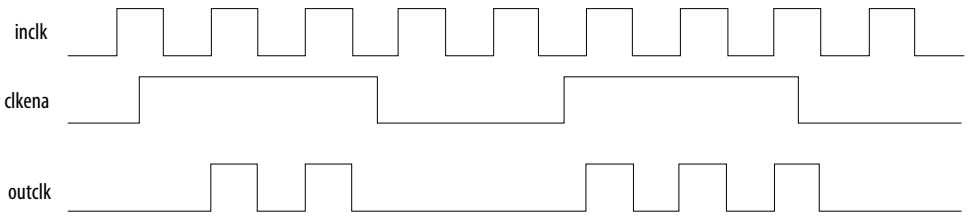
There is one root clock gate per I/O bank and transceiver bank. This gate is a part of the periphery DCM.

#### 2.1.3.1.2. Sector Clock Gate

Every sector of the device has 32 SCLKs. Each SCLK has a clock gate and a path that bypasses the clock gate. The SCLK gates are controlled by clock enable inputs from the core logic. The Intel Quartus Prime software can route up to eight unique clock enable signals to the 32 SCLKs in a sector.

Intel recommends using the clock gate with a negative latch to provide glitch free gating on the output clock signal (`outclk`). The clock gate captures the enable signal (`clkena`) on the next rising edge of the input clock signal (`inclk`). The following timing diagram shows the relationship of the `outclk` with respect to `inclk` and `clkena`.

**Figure 6. Clock Gating Timing Diagram**




The clock signal going into the SCLK network in a sector can only reach the core logic in that sector. When you instantiate a SCLK gate in your design, the Intel Quartus Prime software automatically duplicates the SCLK gate to create a clock gate in every sector to which the clock signal is routed.

The SCLK gate is suitable for cycle-specific clock gating for high-frequency clocks. The timing of the enable path to the SCLK gate is analyzed by the Intel Quartus Prime software.

**Related Information**

- Clock Sector on page 6
  Provides a diagram that shows the dedicated clock resources within a clock sector.
- Clock Control Features on page 8
  Provides a diagram that shows the resources within a SCLK.

#### 2.1.3.1.3. I/O PLL Clock Gate

You can dynamically gate each output counter of the Intel Agilex I/O PLL. This I/O PLL clock gate provides a useful alternative to the root clock gate. The root clock gate can gate only 1 of 7 output counters.

However, the I/O PLL clock gate is not cycle-specific. When you use the I/O PLL clock gate, expect a delay of several clock cycles between the assertion or deassertion of the clock gate and the corresponding change to the clock signal. The number of delay cycles is non-deterministic because the enable signal must be synchronized into the clock domain of the output clock, ensuring a glitch-free gate.

#### 2.1.3.1.4. LAB Clock Gate

The Intel Agilex LAB register has built-in clock gating functionality. The register clock enable mechanism is a hardened data feedback, as shown in the *Clock Gating and Clock Divider in Intel Agilex Clock Network* diagram. The LAB clock gate offers no associated power savings because this is a purely functional clock enable.

The analysis and synthesis phases of the Intel Quartus Prime software infer a LAB clock gate from a behavioral description of clock gating in the register transfer level (RTL). If a physical clock gate is desired, you must instantiate it explicitly.

**Related Information**

Clock Control Features on page 8
Provides the *Clock Gating and Clock Divider in Intel Agilex Clock Network* diagram.

### 2.1.3.2. Clock Divider

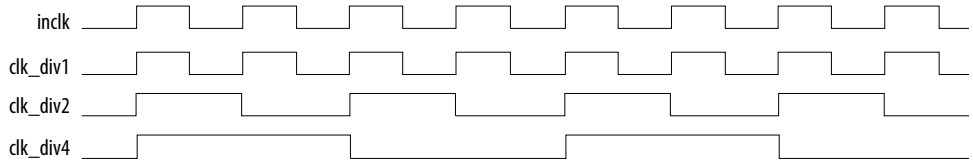
There is one clock divider per I/O bank and transceiver bank. The clock divider is a part of the periphery DCM block and is located close to the root clock gate. The outputs of the clock divider cannot be gated by the root clock gate in the same periphery DCM block. However, this limitation does not apply to the SCLK gate. The clock divider output in the periphery DCM block can drive a SCLK gate after going through the programmable clock routing.

The clock divider has three outputs as follows:

- First output—passes through the input clock.
- Second output—divides the input clock by two.
- Third output—divides the input clock by four.

These three clock outputs are edge-aligned at the output of the clock divider.

**Figure 7. Clock Divider Timing Diagram**




**Related Information**

Clock Control Features on page 8
Provides a diagram that shows the root clock gate and clock divider in the periphery DCM block.

## 2.2. PLLs Architecture and Features

### 2.2.1. PLL Features

**Table 2. PLL Features in Intel Agilex Devices—Preliminary**

| Feature | I/O Bank I/O PLL | Fabric-Feeding I/O PLL |
|---|---|---|
| Integer PLL | Yes | Yes |
| Number of C output counter | 7 | 3 |
| M counter divide factor range | 4 to 160 | 4 to 160 |
| N counter divide factor range | 1 to 110 | 1 to 110 |
| C counter divide factor range | 1 to 512 | 1 to 512 |
| Dedicated external clock outputs (2) | Yes | — |
| Dedicated clock input pins | Yes | Yes |

*continued...*

(2) For dedicated external clock outputs, you must enable access to external clock output port through IOPLL Intel FPGA IP core. There are 2 dedicated external clock output available for each I/O bank I/O PLL.

| Feature | I/O Bank I/O PLL | Fabric-Feeding I/O PLL |
|---|---|---|
| External feedback input pin | Yes | — |
| Source synchronous compensation (3) | Yes | Yes |
| Direct compensation | Yes | Yes |
| Normal compensation (3) | Yes | Yes |
| Zero-delay buffer compensation | Yes | Yes |
| External feedback compensation | Yes | — |
| LVDS compensation | Yes | — |
| Voltage-controlled oscillator (VCO) output drives the DPA clock | Yes | — |
| Phase shift resolution (4) | 78.125 ps | 78.125 ps |
| Programmable duty cycle | Yes | Yes |
| Power down mode | Yes | Yes |
| Bandwidth setting | Low, medium, and high | Medium and high |
| Spread-spectrum input clock tracking (5) | Yes | Yes |

## 2.2.2. PLL Usage

I/O bank I/O PLLs are optimized for use with memory interfaces and LVDS SERDES. You can use both the I/O bank I/O PLLs and fabric-feeding I/O PLLs to:

- Reduce the number of required oscillators on the board
- Reduce the clock pins used in the FPGA by synthesizing multiple clock frequencies from a single reference clock source
- Simplify the design of external memory interfaces and high-speed LVDS interfaces
- Ease timing closure because the I/O PLLs are tightly coupled with the I/Os
- Compensate for clock network delay
- Zero delay buffering

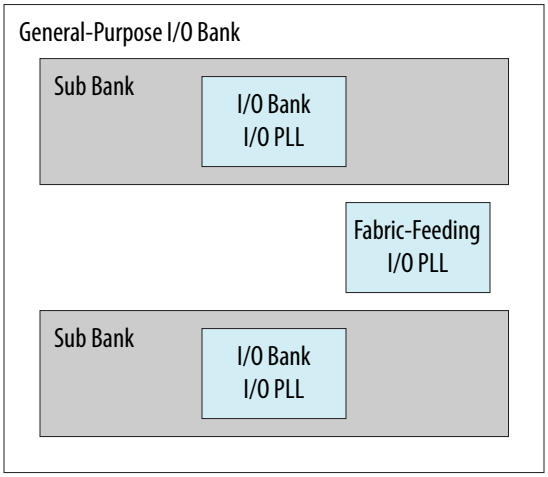
(3) Non-dedicated feedback path option is available for this compensation mode.

(4) The smallest phase shift is determined by the VCO period divided by eight. For degree increments, the Intel Agilex device can shift all output frequencies in increments of at least 45°. Smaller degree increments are possible depending on the frequency and divide parameters.

(5) Provided that input clock jitter is within the input jitter tolerance specifications.

## 2.2.3. PLL Locations

**Figure 8. I/O PLL Locations in I/O Bank**




## 2.2.4. PLL Architecture

**Figure 9. I/O Bank I/O PLL High-Level Block Diagram for Intel Agilex Devices**

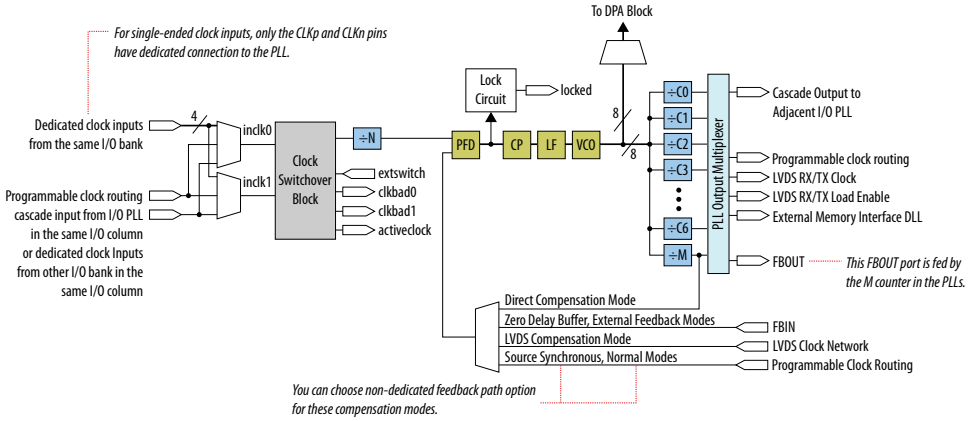
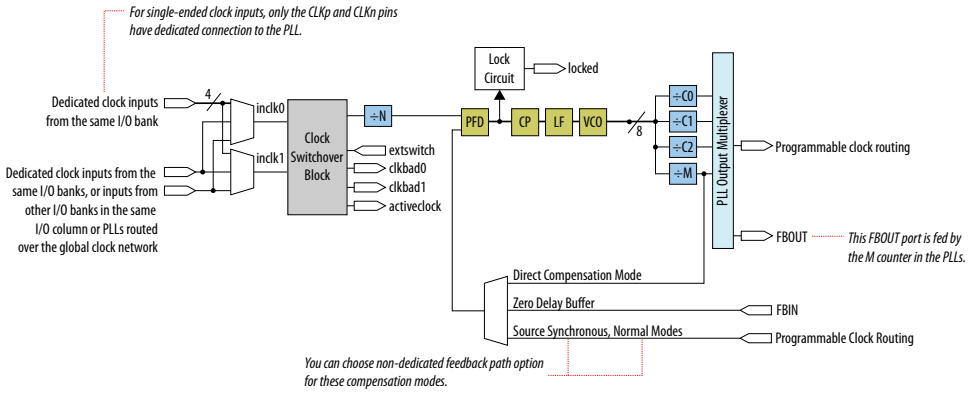



**Figure 10. Fabric-Feeding I/O PLL High-Level Block Diagram for Intel Agilex Devices**



## 2.2.5. PLL Control Signals

You can use the reset signal to control PLL operation and resynchronization, and use the locked signal to observe the status of the PLL.

### 2.2.5.1. Reset

The reset signal port of the IP core for I/O PLL is `reset`.

The reset signal is the reset or resynchronization input for each I/O PLL. The device input pins or internal logic can drive these input signals.

When the reset signal is driven high, the I/O PLL counters reset, clearing the I/O PLL output and placing the I/O PLL out-of-lock. The VCO is then set back to its nominal setting. When the reset signal is driven low again, the I/O PLL resynchronizes to its input clock source as it re-locks.

You must assert the reset signal every time the I/O PLL loses lock to guarantee the correct phase relationship between the I/O PLL input and output clocks. You can set up the I/O PLL to automatically reset (self-reset) after a loss-of-lock condition using the Intel Quartus Prime parameter editor.

You must include the reset signal if either of the following conditions is true:

- I/O PLL reconfiguration or clock switchover is enabled in the design.
- Phase relationships between the I/O PLL input and output clocks must be maintained after a loss-of-lock condition.

*Note:* Reset the I/O PLL after the input clock is stable and within specifications, even when the self-reset feature is enabled, if either one of the following conditions occur:

- The input clock to the I/O PLL is not toggling or is unstable when the FPGA transitions into user mode.
- The I/O PLL cannot lock to the reference clock after reconfiguring the I/O PLL.

**Related Information**

PLL Calibration on page 26

### 2.2.5.2. Locked

The locked signal port of the IP core for the I/O PLL is `locked`.

The lock detection circuit provides a signal to the core logic. The signal indicates when the feedback clock locks onto the reference clock both in phase and frequency.

When PLL loses lock, the output of the PLL starts drifting out of the desired frequency. The downstream logic must be held inactive when PLL has lost lock.

## 2.2.6. PLL Feedback Modes

PLL feedback modes compensate for clock network delays to align the rising edge of the output clock with the rising edge of the PLL's reference clock. Select the appropriate type of compensation for the timing critical clock path in your design.

PLL compensation is not always needed. A PLL should be configured in direct (no compensation) mode unless a need for compensation is identified. Direct mode provides the best PLL jitter performance and avoids expending compensation clocking resources unnecessarily.

The default PLL feedback mode is direct compensation mode.

I/O PLLs support the following PLL feedback modes:

- Direct compensation
- LVDS compensation
- Source synchronous compensation
- Normal compensation
- Zero delay buffer (ZDB) compensation
- External feedback (EFB) compensation

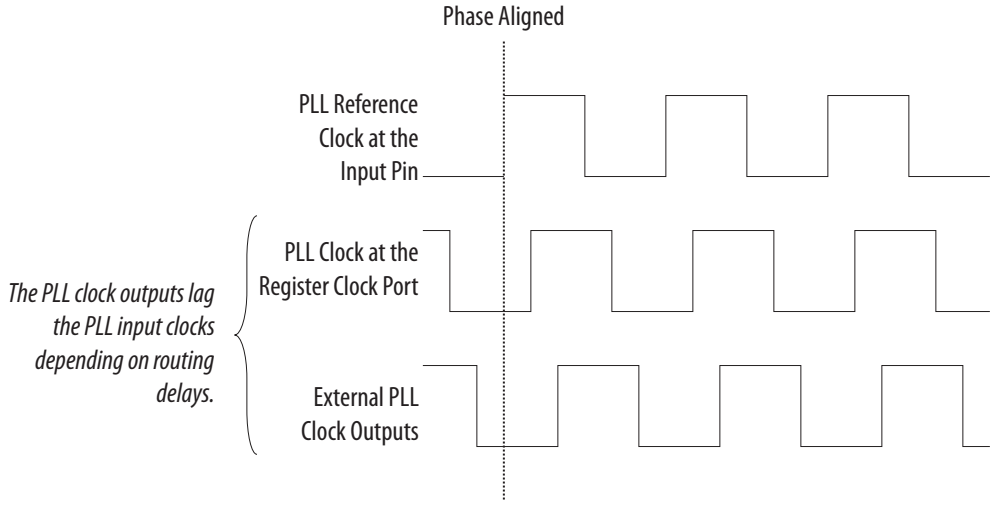
Normal and source synchronous compensation modes compensate for the insertion delay of a routed core clock. For Intel Agilex devices, you can achieve core clock compensation by routing a dedicated feedback clock from the M counter in the I/O PLL to emulate the insertion delay of the compensated C counter output clock network.

Intel recommends the non-dedicated feedback mechanism because it uses the clock resources most efficiently.

### 2.2.6.1. Direct Compensation Mode

In direct mode, the PLL does not compensate for any clock network delays. This mode provides better jitter performance compared to other compensation modes because the clock feedback into the phase frequency detector (PFD) passes through less circuitry. Both the PLL internal- and external-clock outputs are phase-shifted with respect to the PLL clock input.

**Figure 11. Example of Phase Relationship Between the PLL Clocks in Direct Mode**



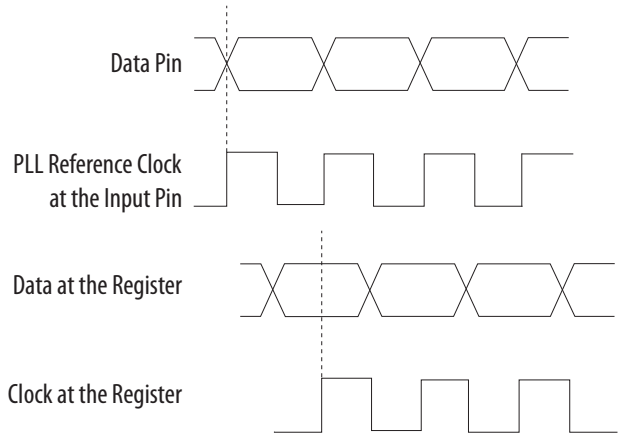
### 2.2.6.2. LVDS Compensation Mode

LVDS compensation mode maintains the same data and clock timing relationship seen at the pins of the internal serializer/deserializer (SERDES) capture register, except that the clock is inverted (180° phase shift). Thus, LVDS compensation mode ideally compensates for the delay of the LVDS clock network, including the difference in delay between the following two paths:

- Data pin-to-SERDES capture register
- Clock input pin-to-SERDES capture register

The output counter must provide the 180° phase shift.

**Figure 12. Example of Phase Relationship Between the Clock and Data in LVDS Compensation Mode**

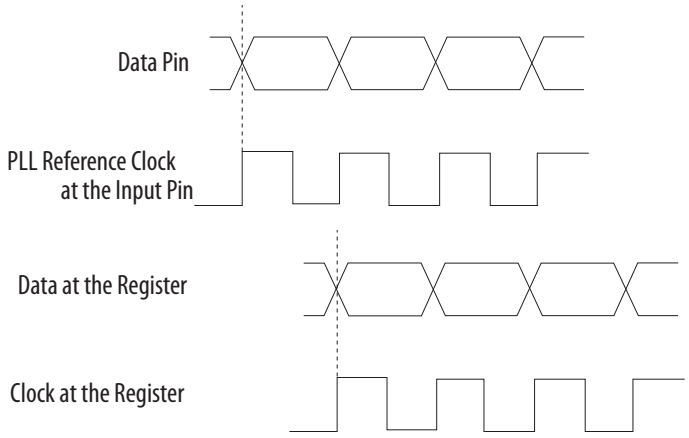



### 2.2.6.3. Source Synchronous Compensation Mode

If the data and clock signals arrive at the same time on the input pins, the same phase relationship is maintained at the clock and data ports of any IOE input register. Data and clock signals at the IOE experience similar buffer delays as long as you use the same I/O standard. Only one output clock can be compensated in source synchronous compensation mode.

Intel recommends source synchronous mode for source synchronous data transfers.

**Figure 13. Example of Phase Relationship Between Clock and Data in Source Synchronous Mode**




The source synchronous mode compensates for the delay of the clock network used and any difference in the delay between the following two paths:

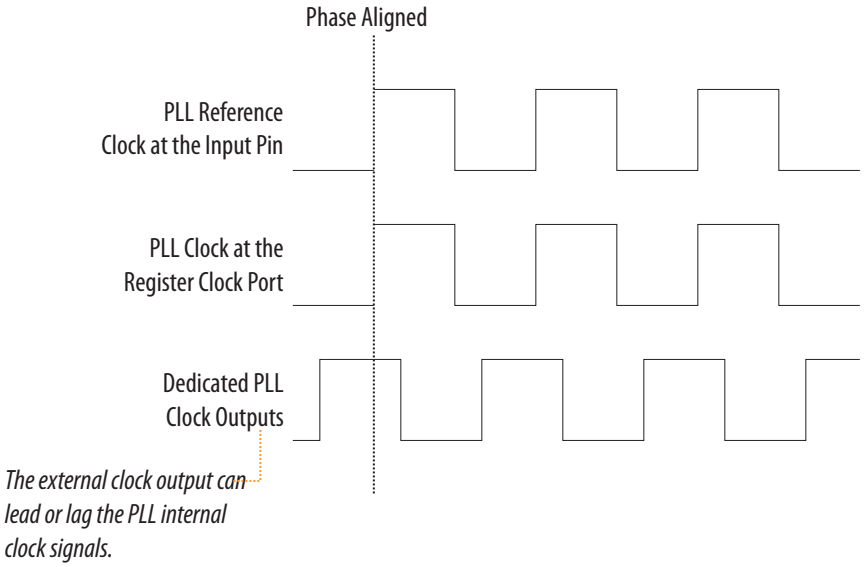
- Data pin to the IOE register input
- Clock input pin to the PLL PFD input

The Intel Agilex PLL can compensate multiple pad-to-input-register paths, such as a data bus when it is set to use source synchronous compensation mode.

### 2.2.6.4. Normal Compensation Mode

An internal clock in normal compensation mode is phase-aligned to the input clock pin. The external clock output pin has a phase delay relative to the clock input pin if connected in this mode. The Intel Quartus Prime Timing Analyzer reports any phase difference between the two. In normal compensation mode, the delay introduced by the clock network is fully compensated. Only one output clock can be compensated in normal compensation mode.

**Figure 14. Example of Phase Relationship Between the PLL Clocks in Normal Compensation Mode**




### 2.2.6.5. Zero-Delay Buffer Mode

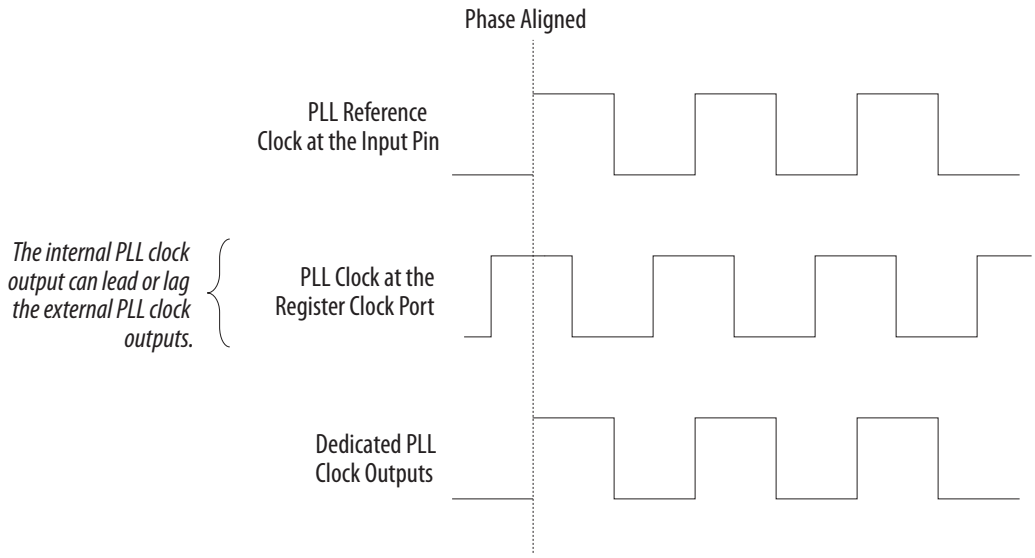
In zero-delay buffer (ZDB) mode, the external clock output pin is phase-aligned with the clock input pin for zero delay through the device.

In this mode, you must use the same I/O standard on the input clocks and clock outputs to guarantee clock alignment at the input and output pins. You cannot use differential I/O standards on the PLL clock input or output pins.

To ensure phase alignment between the `clk` pin and the external clock output (CLKOUT) pin in ZDB mode, instantiate a bidirectional I/O pin in the design. The bidirectional I/O pin serves as the feedback path connecting the `fbout` and `fbin` ports of the PLL. The bidirectional I/O pin must always be assigned a single-ended I/O standard. The PLL uses this bidirectional I/O pin to mimic and compensate for the output delay from the clock output port of the PLL to the external clock output pin.

*Note:* To avoid signal reflection when using ZDB mode, do not place board traces on the bidirectional I/O pin.

**Figure 15. Example of Phase Relationship Between the PLL Clocks in ZDB Mode**



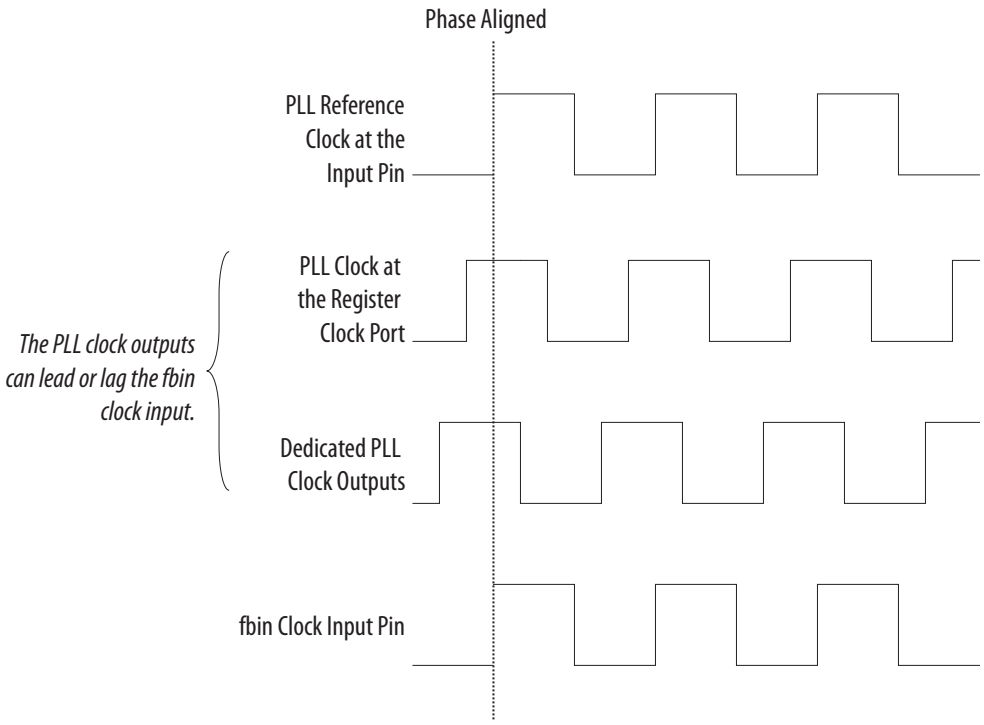

### 2.2.6.6. External Feedback Mode

In external feedback (EFB) mode, the output of the M counter (`fbout`) feeds back to the PLL `fbin` input (using a trace on the board) and becomes part of the feedback loop. EFB mode is only supported for I/O bank I/O PLL, not supported for fabric-feeding I/O PLL.

One of the dual-purpose external clock outputs becomes the `fbin` input pin in this mode. The external feedback input pin, `fbin` is phase-aligned with the clock input pin. Aligning these clocks allows you to remove clock delay and skew between devices.

In EFB mode, you must use the same I/O standard on the input clock, feedback input, and clock outputs.

**Figure 16. Example of Phase Relationship Between the PLL Clocks in EFB Mode**




## 2.2.7. Clock Multiplication and Division

An Intel Agilex PLL output frequency is related to its input reference clock source by the scale factor: M/(N × C) for I/O PLL.

The input clock is divided by a pre-scale factor, N, and is then multiplied by the M feedback factor. The control loop drives the VCO to match $f_{in}$ × (M/N). When using non-dedicated feedback path in normal or source synchronous compensation mode, the control loop drives the VCO to match $f_{in}$ × ((M × $C_i$)/N), where $C_i$ is the compensated `outclk` C counter value. The Intel Quartus Prime software automatically chooses the appropriate scale factors according to the input frequency, multiplication, and division values entered into the Intel FPGA IP cores for I/O PLL.

### Pre-Scale Counter, N and Multiply Counter, M

Each PLL has one pre-scale counter, N, and one multiply counter, M. The M and N counters do not use duty-cycle control because the only purpose of these counters is to calculate frequency division.

### Post-Scale Counter, C

Each output port has a unique post-scale counter, C. For multiple C counter outputs with different frequencies, the VCO is set to the least common multiple of the output frequencies that meets its frequency specifications. For example, if the output frequencies required from one I/O PLL are 55 MHz and 100 MHz, the Intel Quartus

Prime software sets the VCO frequency to 1.1 GHz (the least common multiple of 55 MHz and 100 MHz within the VCO operating frequency range). Then the post-scale counters, C, scale down the VCO frequency for each output port.

#### Integer Mode

The I/O PLL can only operate in integer mode.

## 2.2.8. Programmable Phase Shift

The programmable phase shift feature allows only the I/O PLLs to generate output clocks with a fixed phase offset.

The VCO frequency of the PLL determines the precision of the phase shift. The minimum phase shift increment is 1/8 of the VCO period. For example, if an I/O PLL operates with a VCO frequency of 1000 MHz, phase shift steps of 125 ps are possible.

The Intel Quartus Prime software automatically adjusts the VCO frequency according to the user-specified phase shift values entered into the IP core.

## 2.2.9. Programmable Duty Cycle

The programmable duty cycle feature allows I/O PLLs to generate clock outputs with a variable duty cycle. This feature is only supported by the I/O PLL post-scale counters, C.

The I/O PLL C counter value determines the precision of the duty cycle. The precision is 50% divided by the post-scale counter value. For example, if the C0 counter is 10, steps of 5% are possible for duty-cycle options from 5% to 90%. If the I/O PLL is in external feedback mode, set the duty cycle for the counter driving the fbin pin to 50%.

The Intel Quartus Prime software automatically adjusts the VCO frequency according to the required duty cycle that you enter in the IP core.

Combining the programmable duty cycle with programmable phase shift allows the generation of precise non-overlapping clocks.

## 2.2.10. PLL Cascading

Intel Agilex devices support PLL-to-PLL cascading. You can cascade a maximum of two PLLs. PLL cascading synthesizes more output clock frequencies than a single PLL.

If you cascade PLLs in your design, the source (upstream) PLL must have a low-bandwidth setting, and the destination (downstream) PLL must have a high-bandwidth setting for I/O PLL. During cascading, the output of the source PLL serves as the reference clock (input) of the destination PLL. The bandwidth settings of cascaded PLLs must be different. If the bandwidth settings of the cascaded PLLs are the same, the cascaded PLLs may amplify phase noise at certain frequencies. Intel Agilex devices does not support I/O PLL cascading within the same I/O bank.

Intel Agilex devices support the following PLL-to-PLL cascading modes for I/O bank I/O PLL:

- I/O-PLL-to-I/O-PLL cascading via dedicated cascade path—upstream I/O PLL and downstream I/O PLL must be in the same I/O column.
- I/O-PLL-to-I/O-PLL cascading via core clock fabric—no restriction on locations of upstream and downstream I/O PLL.

The `permit_cal` input of the downstream I/O PLL must be connected to the `locked` output of the upstream I/O PLL in both PLL cascading modes.

The following figures show the connectivity required between the upstream and downstream I/O PLL for both the PLL cascading modes.

**Figure 17. I/O-PLL-to-I/O-PLL Cascading Via Dedicated Cascade Path**

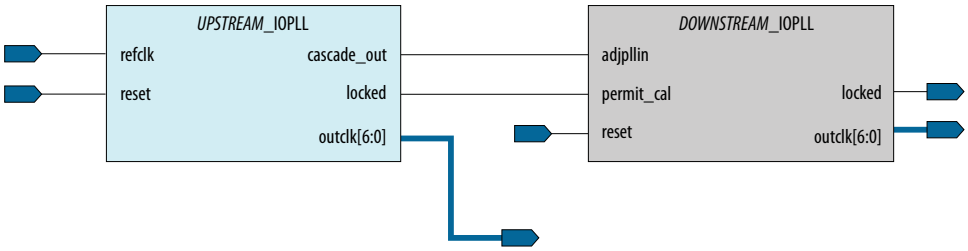
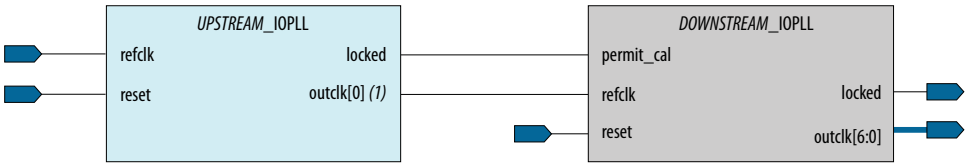



**Figure 18. I/O-PLL-to-I/O-PLL Cascading Via Core Clock Fabric**




Note:
(1) You may connect any of the outclk from the upstream I/O PLL to the refclk port in the downstream I/O PLL when cascading the I/O PLL via core clock fabric.

## 2.2.11. PLL Input Clock Switchover

The clock switchover feature allows the I/O PLL to switch between two reference input clocks. Use this feature for clock redundancy or for a dual-clock domain application where a system turns to the redundant clock if the previous clock stops running. The design can perform clock switchover automatically when the clock is no longer toggling or based on a user control signal, `extswitch`.

Intel Agilex I/O PLLs support the following clock switchover modes:

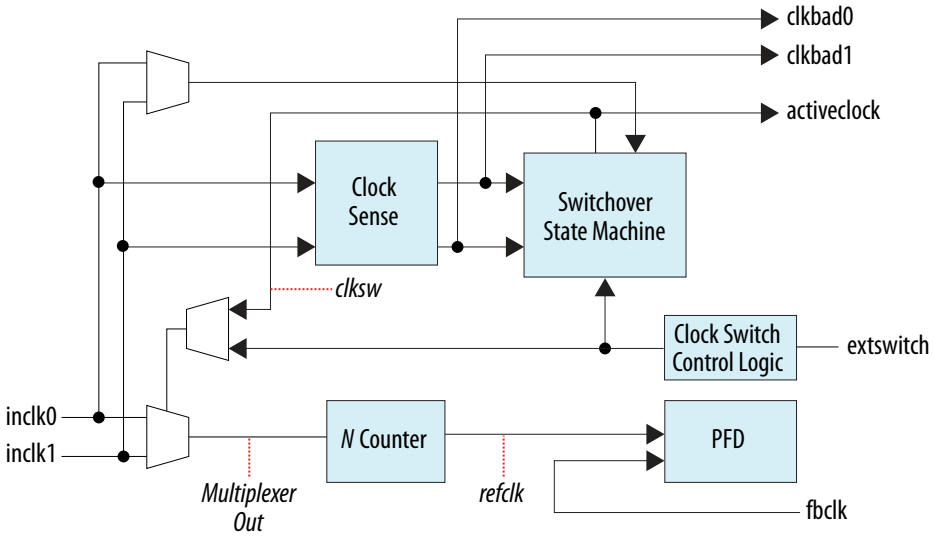
- Automatic switchover—the clock sense circuit monitors the current reference clock. If the current reference clock stops toggling, the reference clock automatically switches to `inclk0` or `inclk1` clock.
- Manual clock switchover—clock switchover is controlled using the `extswitch` signal. When the `extswitch` signal goes from logic high to logic low, and stays low for at least three clock cycles for the `inclk` being switched to, the reference clock to the I/O PLL is switched from `inclk0` to `inclk1`, or vice-versa.
- Automatic switchover with manual override—this mode combines automatic switchover and manual clock switchover. When the `extswitch` signal goes low, it overrides the automatic clock switchover function. As long as the `extswitch` signal is low, further switchover action is blocked.

### 2.2.11.1. Automatic Switchover

Intel Agilex I/O PLLs support a fully configurable clock switchover capability.

**Figure 19. Automatic Clock Switchover Circuit Block Diagram**

This figure shows a block diagram of the automatic switchover circuit built into the I/O PLL.




When the current reference clock is not present, the clock sense block automatically switches to the backup clock for I/O PLL reference. You can select a clock source as the backup clock by connecting it to the `inclk1` port of the I/O PLL in your design.

The clock switchover circuit sends out three status signals—`clkbad0`, `clkbad1`, and `activeclock`—from the I/O PLL to implement a custom switchover circuit in the logic array.

In automatic switchover mode, the `clkbad0` and `clkbad1` signals indicate the status of the two clock inputs. When they are asserted, the clock sense block detects that the corresponding clock input has stopped toggling. These two signals are not valid if the frequency difference between `inclk0` and `inclk1` is greater than 20%.

The `activeclock` signal indicates which of the two clock inputs (`inclk0` or `inclk1`) is being selected as the reference clock to the I/O PLL. When the frequency difference between the two clock inputs is more than 20%, the `activeclock` signal is the only valid status signal.

Use the switchover circuitry to automatically switch between `inclk0` and `inclk1` when the current reference clock to the I/O PLL stops toggling. You can switch back and forth between `inclk0` and `inclk1` any number of times when one of the two clocks fails and the other clock is available.

For example, in applications that require a redundant clock with the same frequency as the reference clock, the switchover state machine generates a signal (`clksw`) that controls the multiplexer select input. In this case, `inclk1` becomes the reference clock for the I/O PLL.

When using automatic clock switchover mode, the following requirements must be satisfied:

- Both clock inputs must be running when the FPGA is configured.
- The period of the two clock inputs can differ by no more than 20%.
- The input clocks must meet the input jitter specifications and I/O standard specifications.

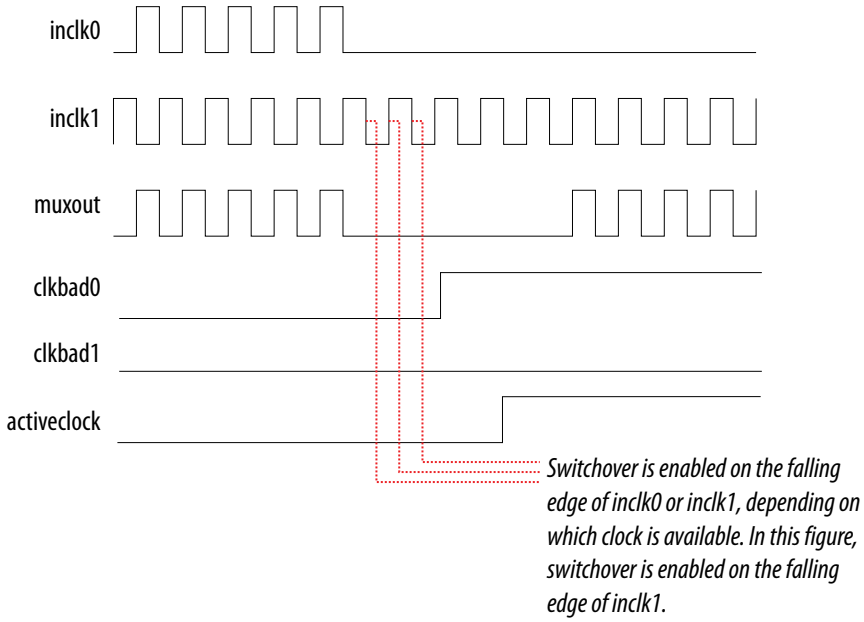
Glitches in the input clock may be seen as a greater than 20% difference in frequency between the input clocks.

If the current clock input stops toggling while the other clock is also not toggling, switchover is not initiated and the `clkbad[0..1]` signals are not valid. If both clock inputs are not the same frequency, but their period difference is within 20%, the clock sense block detects when a clock stops toggling. However, the I/O PLL may lose lock after the switchover is completed and needs time to relock.

*Note:* You must reset the I/O PLL using the reset signal to maintain the phase relationships between the I/O PLL input and output clocks when using clock switchover.

**Figure 20. Automatic Switchover After Loss of Clock Detection**

This figure shows an example waveform of the switchover feature in automatic switchover mode. In this example, the `inclk0` signal is held low. After the `inclk0` signal is held low for approximately two clock cycles, the clock sense circuitry drives the `clkbad0` signal high. As the reference clock signal (`inclk0`) is not toggling, the switchover state machine controls the multiplexer through the `extswitch` signal to switch to the backup clock, `inclk1`.




## 2.2.11.2. Automatic Switchover with Manual Override

In automatic switchover with manual override mode, you can use the `extswitch` signal for user- or system-controlled switch conditions. You can use this mode for same-frequency switchover, or to switch between inputs of different frequencies.

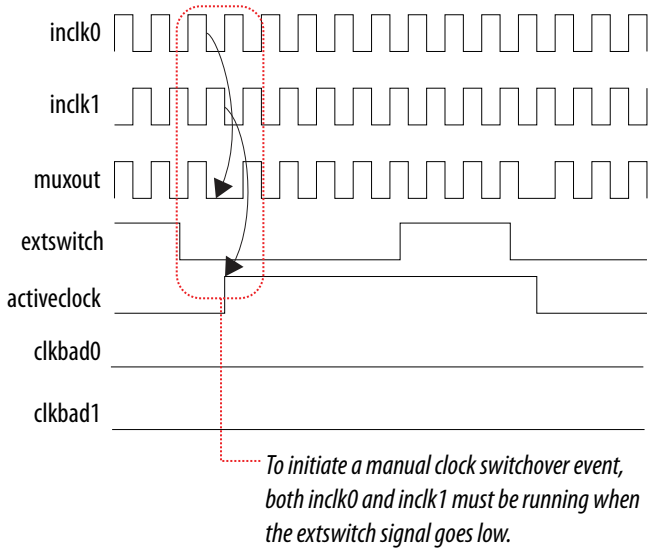
For example, if `inclk0` is 66 MHz and `inclk1` is 200 MHz, you must control switchover using the `extswitch` signal. The automatic clock-sense circuitry cannot monitor clock input (`inclk0` and `inclk1`) frequencies with a frequency difference of more than 100% (2×).

This feature is useful when the clock sources originate from multiple cards on the backplane, requiring a system-controlled switchover between the frequencies of operation.

You must choose the backup clock frequency and set the `M`, `N`, and `C` counters so that the VCO operates within the recommended operating frequency range. The Intel Quartus Prime software notifies you if a given combination of `inclk0` and `inclk1` frequencies cannot meet this requirement.

**Figure 21. Clock Switchover Using the `extswitch` (Manual) Control**

This figure shows a clock switchover waveform controlled by the `extswitch` signal. In this case, both clock sources are functional and `inclk0` is selected as the reference clock. The switchover sequence starts when the `extswitch` signal goes low. On the falling edge of `inclk0`, the counter's reference clock, `muxout`, is gated off to prevent clock glitching. On the falling edge of `inclk1`, the reference clock multiplexer switches from `inclk0` to `inclk1` as the I/O PLL reference. The `activeclock` signal changes to indicate the clock which is currently feeding the I/O PLL.




In automatic override with manual switchover mode, the `activeclock` signal inverts after the `extswitch` signal transitions from logic high to logic low. Since both clocks are still functional during the manual switch, neither `clkbad` signal goes high. Because the switchover circuit is negative-edge sensitive, the rising edge of the `extswitch` signal does not cause the circuit to switch back from `inclk1` to `inclk0`. When the `extswitch` signal goes low again, the process repeats.

The `extswitch` signal and automatic switch work only if the clock being switched to is available. If the clock is not available, the state machine waits until the clock is available.

### 2.2.11.3. Manual Clock Switchover

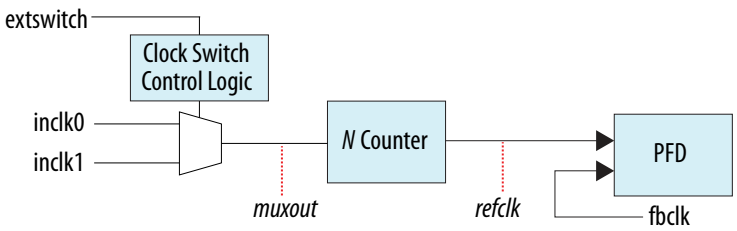
In manual clock switchover mode, the `extswitch` signal controls whether `inclk0` or `inclk1` is selected as the input clock to the I/O PLL. By default, `inclk0` is selected.

A clock switchover event is initiated when the `extswitch` signal transitions from logic high to logic low, and is held low for at least three `inclk` cycles for the `inclk` clock being switched to.

You must bring the `extswitch` signal back high again to perform another switchover event. If you do not require another switchover event, you can leave the `extswitch` signal in a logic low state after the initial switch.

If `inclk0` and `inclk1` are different frequencies and are always running, the `extswitch` signal minimum low time must be greater than or equal to three of the slower frequency `inclk0` and `inclk1` cycles.

**Figure 22. Manual Clock Switchover Circuitry in Intel Agilex I/O PLLs**




You can delay the clock switchover action by specifying the switchover delay in the Intel FPGA IP cores for the I/O PLL. When you specify the switchover delay, the `extswitch` signal must be held low for at least three `inclk` cycles for the `inclk` being switched to plus the number of the delay cycles that you specify to initiate a clock switchover.

## 2.2.12. PLL Reconfiguration and Dynamic Phase Shift

Intel Agilex devices support PLL reconfiguration and dynamic phase shift with the following features:

- PLL reconfiguration—I/O PLL can reconfigure the `M`, `N`, `C` counters, and bandwidth setting.
- Dynamic phase shift—I/O PLL can perform positive or negative phase shift. Able to shift multiple phase steps each time, where one phase step is equal to 1/8 of the VCO period.

## 2.2.13. PLL Calibration

I/O PLLs include both analog and digital blocks that require calibration to compensate for process, voltage, and temperature (PVT) variations. Intel Agilex uses the I/O manager to perform calibration routines.

There are two main types of calibration.

- Power-up calibration—initiates automatically at device power-up and runs during device configuration.
- User calibration—if you perform dynamic reconfiguration or change the reference clock frequency of the I/O PLL, you must perform user recalibration. You must enable the required calibration sequence.

To successfully complete the calibration process, `OSC_CLK_1` clocks and all reference clocks driving the I/O PLLs must be stable and free running at the start of FPGA configuration. If clock switchover is enabled, both reference clocks must be present for calibration. During user mode, when the I/O PLL does not detect a reference clock during configuration, calibration attempts continue periodically. After calibration has completed, the I/O PLL is locked automatically.

### 2.2.13.1. Power-Up Calibration

After device power-up, the I/O manager automatically initiates the calibration process. The process continues during device programming.

### 2.2.13.2. User Calibration

The I/O PLL must be recalibrated for any of the following conditions after device power up:

- Dynamic I/O PLL reconfiguration that changes the `M` or `N` counter settings is performed.
- Change of the reference clock frequency to the I/O PLL.

Recalibration is not necessary when using clock switchover to a secondary reference clock with a different frequency than the primary reference clock. The I/O PLL stores the calibration settings for both reference clocks after power-up calibration.

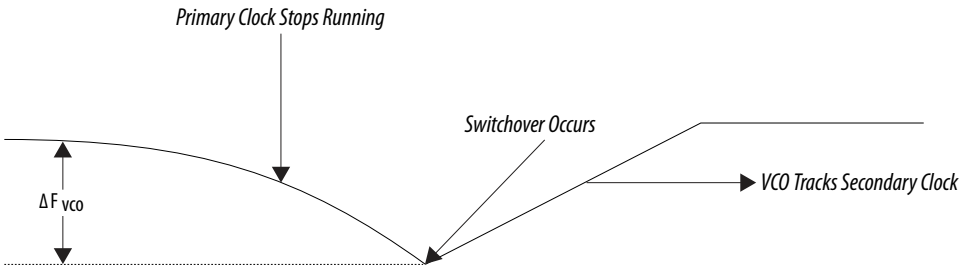
# 3. Intel Agilex Clocking and PLL Design Considerations

## 3.1. Guideline: Clock Switchover

When implementing clock switchover in Intel Agilex I/O PLLs, refer to the following guidelines:

- Automatic clock switchover requires that the `inclk0` and `inclk1` frequencies are within 20% of each other. Failing to meet this requirement causes the `clkbad0` and `clkbad1` signals to not function properly.
- When using manual clock switchover, the difference between `inclk0` and `inclk1` can be more than 100% (2×). However, differences in frequency, phase, or both, of the two clock sources may cause the I/O PLL to lose lock. Resetting the I/O PLL ensures that you maintain the correct phase relationships between the input and output clocks.
- Both `inclk0` and `inclk1` must be running when the `extswitch` signal goes low to initiate the manual clock switchover event. Failing to meet this requirement causes the clock switchover to not function properly.
- Applications that require a clock switchover feature and a small frequency drift must use a low-bandwidth I/O PLL. When referencing input clock changes, the low-bandwidth I/O PLL reacts more slowly than a high-bandwidth I/O PLL. When switchover happens, a low-bandwidth I/O PLL propagates the stopping of the clock to the output more slowly than a high-bandwidth I/O PLL. However, the low-bandwidth I/O PLL also increases lock time.
- After a switchover occurs, there may be a finite resynchronization period for the I/O PLL to lock onto a new clock. The time it takes for the I/O PLL to relock depends on the I/O PLL configuration.
- If the phase relationship between the input clock to the I/O PLL and the output clock from the I/O PLL is important in your design, assert the reset signal for at least 10 ns after performing a clock switchover. Wait for the locked signal to go high and be stable before re-enabling the output clocks from the I/O PLL.
- The VCO frequency gradually decreases when the current clock is lost and then increases as the VCO locks on to the backup clock, as shown in the following figure.

**Figure 24. VCO Switchover Operating Frequency**




## 3.2. Guideline: Timing Closure

For timing closure, refer to the following guidelines:

- Reconfiguring a PLL's counter and loop filter settings changes both the output frequency and the clock uncertainty of that I/O PLL. Dynamic phase shift only affects the output clock phase.
- The Timing Analyzer in the Intel Quartus Prime software performs timing analysis for the initial PLL settings only. You must verify that your design closes timing after dynamic reconfiguration or dynamic phase shift.
- Intel recommends compiling the I/O PLL designs with each intended configuration setting to determine the variation in the clock with the I/O PLL settings.

### Related Information

Creating Clocks and Clock Constraints chapter, Intel Quartus Prime Pro Edition User Guide: Timing Analyzer

## 3.3. Guideline: Resetting the PLL

To reset the PLL, refer to the following guidelines:

- When changing the M counter, N counter, or loop filter settings, the I/O PLL may lose and regain lock. To maintain the appropriate phase relationship between the reference clock and output clocks, assert the `areset` signal to reset the I/O PLL after reconfiguration is complete. Intel recommends always resetting the I/O PLL after any reconfiguration operation to the M counter, N counter, or loop filter settings.
- When changing the C counter settings, you may lose the expected phase relationship between the C counters. Assert the `areset` signal after reconfiguration is complete to restore the expected phase relationship. Reset is not required if the phase relationships are not important to your application.
- Resetting the I/O PLL does not modify the counter or loop filter settings. However, resetting the I/O PLL undoes any dynamic phase shift operations that were performed. After the I/O PLL is reset, the phase shift on the C counters is restored to the originally programmed settings.

## 3.4. Guideline: Configuration Constraints

The I/O PLL configuration must obey the following constraints:

- The phase frequency detector (PFD) and VCO each have a legal frequency range of operation.
- The loop filter settings must be appropriate for the M counter value and user-selected bandwidth mode.

If any of these configuration constraints are violated, the I/O PLL may fail to lock or may exhibit poor jitter performance.

## 3.5. Guideline: I/O PLL Reconfiguration

To reconfigure the I/O PLL, refer to the following guidelines:

- If the reference clock frequency changes, you must recalibrate the I/O PLL using the IP core.
- The I/O PLL reconfiguration interface must have a free running `mgmt_clk` signal. The `mgmt_clk` signal must be less than 100 MHz. This interface eliminates the need to precisely control the start and stop of `mgmt_clk` signal.
- The I/O PLL can be reconfigured with `.mif` streaming mode and advanced mode using the IP core. Intel recommends using the `.mif` streaming mode.
- Use caution when reconfiguring an I/O PLL with a non-zero phase shift setting. Modifying the M counter or N counter settings does not change the relative phase shift (in percent), but alters the absolute phase shift (in picoseconds). Modifying the C counter settings does not change the absolute phase shift, but modifies the relative phase shift.

## 3.6. Clocking Constraints

The common clocking constraint commands that are commonly used for clock and PLLs in `.sdc` file are as follows:

- `create_clock`
- `create_generated_clock`
- `create_clock_uncertainty`

## 3.7. IP Core Constraints

To implement the IOPLL IP core, you must adhere to the following constraints:

- Any SDC design constraints referring to the I/O PLL clocks must be listed after the SDC constraints for the IOPLL IP core.
- Intel recommends reading the SDC for all I/O PLLs first in a design. You can do this by listing the IP before others in the `.qsf` file.

# 4. Clock Control Intel FPGA IP Core

The Clock Control IP core provides clock control features such as enabling entry to the clock network, clock multiplexing, clock gating, and clock division for the Intel Agilex devices.

## 4.1. Release Information for Clock Control Intel FPGA IP

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

The IP versioning scheme (X.Y.Z) number changes from one software version to another. A change in:

- X indicates a major revision of the IP. If you update your Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

**Table 3. Clock Control Intel FPGA IP Core Current Release Information**

| Item | Description |
|---|---|
| IP Version | 1.0.0 |
| Intel Quartus Prime Version | 19.3 |
| Release Date | 2019.09.30 |

## 4.2. Clock Control IP Core Parameters

**Table 4. Clock Control IP Core Parameters for Intel Agilex Devices**

| Parameter | Legal Value | Description |
|---|---|---|
| **Number of Clock Inputs** | **1**, **2**, or **4** | Specify the number of input clock sources for the clock control block. You can specify up to four clock inputs.<br>Clock multiplexing in Intel Agilex devices is implemented using soft logic in the core. |
| **Clock Enable** | On or Off | Turn on this option if you want to gate your clock output with an enable signal. This option disables the option to use clock division. |
| **Clock Enable Type** | **Root Level** or **Distributed Sector Level** | Select the clock gates located in the periphery or the gates located in the sector. For more information about the clock gates, refer to the Clock Gating section. |

*continued...*

| Parameter | Legal Value | Description |
|---|---|---|
| **Enable Register Mode** | **Negative Latch** or **None** | Specify if the enable signal should be latched. |
| **Clock Divider** | On or Off | Turn on this option if you want to use the clock division block in the periphery. |
| **Clock Divider Output Ports** | **Divide 1x**, **Divide 1x and 2x**, or **Divide 1x, 2x and 4x** | Specify the combination of passing your clock through, dividing your clock by 2, or dividing your clock by 4. |

## 4.3. Clock Control IP Core Ports and Signals

### Table 5. Clock Control IP Core Ports for Intel Agilex Devices

| Port Name | Description |
|---|---|
| `inclk` | Input signal to the clock network. |
| `inclk0x`, `inclk1x`, `inclk2x`, `inclk3x` | Input signals to the clock network based on the value selected for the **Number of Clock Inputs** parameter. |
| `clkselect[]` | Input that dynamically selects the clock source to drive the clock network that is driven by the clock buffer.<br>Input port `[1 DOWNTO 0]` wide.<br>The following list shows the signal selection for the `clkselect[]` value:<br>• `2'b00` selects `inclk0x`<br>• `2'b01` selects `inclk1x`<br>• `2'b10` selects `inclk2x`<br>• `2'b11` selects `inclk3x` |
| `outclk` | Output of the Clock Control IP core when **Clock Divider** option is not selected. |
| `ena` | Clock enable of the clock gate block. This signal is active-high. |
| `clock_div1x`, `clock_div2x`, `clock_div4x` | Outputs of the Clock Control IP core when the **Clock Divider** option is selected. The exact combination of ports exposed depends on the value specified for the **Clock Divider Output Ports** parameter.<br>• `clock_div1x` is the same as `inclk`<br>• `clock_div2x` divides `inclk` by 2<br>• `clock_div4x` divides `inclk` by 4 |

# 5. IOPLL Intel FPGA IP Core

The IOPLL IP core allows you to configure the settings of the Intel Agilex I/O PLL.

The IOPLL IP core supports the following features:

- Supports six different clock feedback modes: direct, external feedback, normal, source synchronous, zero delay buffer, and LVDS mode.
- Generates up to seven output clocks for I/O bank I/O PLL and three output clocks for fabric-feeding I/O PLL for the Intel Agilex device.
- Switches between two reference input clocks.
- Supports adjacent PLL (`adjpllin`) input to connect with an upstream PLL in PLL dedicated cascading mode.
- Generates the Memory Initialization File (**.mif**) and allows PLL dynamic reconfiguration.
- Supports PLL dynamic phase shift.

## 5.1. Release Information for IOPLL Intel FPGA IP

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

The IP versioning scheme (X.Y.Z) number changes from one software version to another. A change in:

- X indicates a major revision of the IP. If you update your Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

**Table 6. IOPLL Intel FPGA IP Core Current Release Information**

| Item | Description |
|---|---|
| IP Version | 19.3.0 |
| Intel Quartus Prime Version | 19.3 |
| Release Date | 2019.09.30 |

## 5.2. .mif File Generation

You can generate the `.mif` files in the IOPLL IP core parameter editor. You must use the `.mif` file with the IOPLL Reconfig Intel FPGA IP core.

**Related Information**

IOPLL Reconfig Intel FPGA IP Core on page 41

### 5.2.1. Generating a New .mif File

To generate a new `.mif` file containing a single I/O PLL configuration, follow these steps:

1. At the **Dynamic Reconfiguration** tab, select **Enable dynamic reconfiguration of PLL**.
2. For **MIF Generation Options**, select **Generate New MIF File**.
3. For **Path of New MIF file**, specify a file name.
4. For **Name of Current Configuration**, specify the name of the current configuration of the I/O PLL.
5. Click **Create MIF File**.

### 5.2.2. Adding Configurations to Existing .mif File

You can append new configurations to an existing `.mif` file. To store more configurations in a `.mif` file, follow these steps:

1. At the **Dynamic Reconfiguration** tab, select **Enable dynamic reconfiguration of PLL**.
2. For **MIF Generation Options**, select **Add Configuration to Existing MIF File**.
3. For **Path of New MIF file**, specify a file name.
4. For **Name of Current Configuration**, specify the name of the new configuration of the I/O PLL.
5. Click **Append to MIF File**.

## 5.3. IOPLL IP Core Parameters

The IOPLL IP core parameter editor appears in the PLL category of the IP Catalog.

### 5.3.1. IOPLL IP Core Parameters - PLL Tab

**Table 7. IOPLL IP Core Parameters - PLL Tab for Intel Agilex Devices**

| Parameter | Legal Value | Description |
|---|---|---|
| **Device Family** | Intel Agilex | Specifies the device family. |
| **Component** | — | Specifies the targeted device. |
| **Speed Grade** | — | Specifies the speed grade for targeted device. |

*continued...*

| Parameter | Legal Value | Description |
|---|---|---|
| **IOPLL Type** | Fabric-Feeding, I/O Bank | Select the type of the I/O PLL. The fabric-feeding I/O PLL locations have fewer `outclks` than the I/O bank I/O PLLs, and cannot be used as External I/O PLL in the LVDS SERDES Intel FPGA IP core. |
| **Reference Clock Frequency** | — | Specifies the input frequency for the input clock, `refclk`, in MHz. The default value is **100.0 MHz**. The minimum and maximum value is dependent on the selected device. |
| **Refclk source is global clock** | Turn on or Turn off | Specifies whether the reference clock source is a global clock. Intel recommends using a dedicated reference clock pin instead of a global clock to minimize jitter. If a global reference clock source is required, this clock must still be promoted using the Assignment Editor. |
| **Enable Locked Output Port** | Turn on or Turn off | Turn on to enable the `locked` port. |
| **Enable physical output clock parameters** | Turn on or Turn off | Turn on to enter physical PLL counter parameters instead of specifying a desired output clock frequency. |
| **Compensation Mode** | **direct**, **external feedback**(6), **normal**, **source synchronous**, **zero delay buffer**, or **lvds**(6) | Specifies the operation of the PLL. The default operation is **direct** mode.<br>• If you select the **direct** mode, the PLL minimizes the length of the feedback path to produce the smallest possible jitter at the PLL output. The internal-clock and external-clock outputs of the PLL are phase-shifted with respect to the PLL clock input. In this mode, the PLL does not compensate for any clock networks.<br>• If you select the **external feedback** mode, you must connect the `fbclk` input port to an input pin. A board-level connection must connect both the input pin and external clock output port, `fboutclk`. The `fbclk` port is aligned with the input clock.<br>• If you select the **normal** mode, the PLL compensates for the delay of the internal clock network used by the clock output. If the PLL is also used to drive an external clock output pin, a corresponding phase shift of the signal on the output pin occurs.<br>• If you select the **source synchronous** mode, the clock delay from pin to I/O input register matches the data delay from pin to I/O input register.<br>• If you select the **zero delay buffer** mode, the PLL must feed an external clock output pin and compensate for the delay introduced by that pin. The signal observed on the pin is synchronized to the input clock. The PLL clock output connects to the `altbidir` port and drives `zdbfbclk` as an output port. If the PLL also drives the internal clock network, a corresponding phase shift of that network occurs.<br>• If you select the **lvds** mode, the same data and clock timing relationship of the pins at the internal SERDES capture register is maintained. The mode compensates for the delays in LVDS clock network, and between the data pin and clock input pin to the SERDES capture register paths. |
| **Compensated Outclk** (7) | 0–6 | Allows you to select which output clock (`outclk`) to be compensated. The feedback mode compensates for the clock network delay of the `outclk` selected. This feedback mode ensures correct phase relationship between I/O PLL input and output clocks only for the selected `outclk`. |
| **Use Nondedicated Feedback Path** (7) | Turn on or Turn off | Turn on to conserve clock resources and improve timing analysis. However, this feature creates frequency limitations and disables phase shift. |

***continued...***

(6) This option is only available when selecting **I/O Bank** on the **IOPLL Type**.

(7) This option is only available when either **normal** or **source synchronous** mode is selected.

| Parameter | Legal Value | Description |
|---|---|---|
| **Number of Clocks** | 1–3 (fabric-feeding), 1–7 (I/O bank) | Specifies the number of output clocks required for each device in the PLL design. The requested settings for output frequency, phase shift, and duty cycle are shown based on the number of clocks selected. |
| **Multiply Factor (M-Counter)** (8) | 4–160 | Specifies the multiply factor of M-counter. |
| **Divide Factor (N-Counter)** (8) | 1–110 | Specifies the divide factor of N-counter. |
| **Specify VCO Frequency** | Turn on or Turn off | Allows you to restrict the VCO frequency to the specified value. This is useful when creating a PLL for LVDS external mode, or if a specific dynamic phase shift step size is desired. |
| **Desired VCO Frequency** (9) | — | Specifies the VCO frequency for the PLL in MHz. The default value is **600.0 MHz**. |
| **Actual VCO Frequency** | — | • When **Enable physical output clock parameters** is turned on—displays the VCO frequency based on the values for **Reference Clock Frequency**, **Multiply Factor (M-Counter)**, and **Divide Factor (N-Counter)**.<br>• When **Enable physical output clock parameters** is turned off and **Specify VCO frequency** is turned on—allows you to specify the requested value for the VCO frequency. The default value is **600.0 MHz**. |
| **Give clock global name** | Turn on or Turn off | Allows you to rename the output clock name. |
| **Clock Name** | — | The user clock name for Synopsis Design Constraints (SDC). |
| **Divide Factor (C-Counter)** (8) | 1-512 | Specifies the divide factor for the output clock (C-counter). |
| **Desired Frequency** | — | Specifies the output clock frequency of the corresponding output clock port, `outclk[]`, in MHz. The default value is **100.0 MHz**. The minimum and maximum values depend on the device used. The PLL only reads the numerals in the first six decimal places. |
| **Actual Frequency** | — | Allows you to select the actual output clock frequency from a list of achievable frequencies. The default value is the closest achievable frequency to the desired frequency. |
| **Phase Shift units** | **ps** or **degrees** | Specifies the phase shift unit for the corresponding output clock port, `outclk[]`, in picoseconds (ps) or degrees. |
| **Desired Phase Shift** | — | Specifies the requested value for the phase shift. The default value is **0 ps**. |
| **Actual Phase Shift** | — | Allows you to select the actual phase shift from a list of achievable phase shift values. The default value is the closest achievable phase shift to the desired phase shift. |
| **Desired Duty Cycle** | 0.0–100.0 | Specifies the requested value for the duty cycle. The default value is **50.0%**. |
| **Actual Duty Cycle** | — | Allows you to select the actual duty cycle from a list of achievable duty cycle values. The default value is the closest achievable duty cycle to the desired duty cycle. |

(8) This parameter is only available when **Enable physical output clock parameters** is turned on.

(9) This parameter is only available when **Specify VCO Frequency** is turned on and **Enable physical output clock parameters** is turned off.

## 5.3.2. IOPLL IP Core Parameters - Settings Tab

**Table 8. IOPLL IP Core Parameters - Settings Tab for Intel Agilex Devices**

| Parameter | Legal Value | Description |
|---|---|---|
| **PLL Bandwidth Preset** | **Low**(10), **Medium**, or **High** | Specifies the PLL bandwidth preset setting. The default selection is **Medium** for fabric-feeding I/O PLL, and **Low** for I/O bank I/O PLL. |
| **Lock Threshold Setting** | **Low Lock Time**, **Medium Lock Time**, or **High Lock Time** | This setting determines the sensitivity of the I/O PLL when detecting lock. This is a tradeoff between the time it takes to lock and the accuracy of the `outclk` frequency when `locked` is first asserted. For applications that require the I/O PLL to lock quickly, **Low Lock Time** is the best option.<br>The estimated lock times are 30 µs + *a* × `refclk_period`, where *a* is 100, 2048, and 4095 for **Low Lock Time**, **Medium Lock Time**, and **High Lock Time** respectively. |
| **PLL Auto Reset** | Turn on or Turn off | Automatically self-resets the PLL on loss of lock. |
| **Create a second input clk 'refclk1'** | Turn on or Turn off | Turn on to provide a backup clock attached to your PLL that can switch with your original reference clock. |
| **Second Reference Clock Frequency** (11) | — | Selects the frequency of the second input clock signal. The default value is **100.0 MHz**. The minimum and maximum value is dependent on the device used. |
| **Create an 'active_clk' signal to indicate the input clock in use** (11) | Turn on or Turn off | Turn on to create the `activeclk` output. The `activeclk` output indicates the input clock which is in use by the PLL. Output signal low indicates `refclk` and output signal high indicates `refclk1`. |
| **Create a 'clkbad' signal for each of the input clocks** (11) | Turn on or Turn off | Turn on to create two `clkbad` outputs, one for each input clock. Output signal low indicates the clock is working and output signal high indicates the clock is not working. |
| **Switchover Mode** (11) | **Automatic Switchover**, **Manual Switchover**, or **Automatic Switchover with Manual Override** | Specifies the switchover mode for design application. The IP supports three switchover modes:<br>• If you select the **Automatic Switchover** mode, the PLL circuitry monitors the selected reference clock. If one clock stops, the circuit automatically switches to the backup clock in a few clock cycles and updates the status signals, `clkbad` and `activeclk`.<br>• If you select the **Manual Switchover** mode, when the control signal, `extswitch`, changes from logic low to logic high, and stays high for at least three clock cycles, the input clock switches to the other clock. The `extswitch` can be generated from FPGA core logic or input pin.<br>• If you select **Automatic Switchover with Manual Override** mode, when the `extswitch` signal is high, it overrides the automatic switch function. As long as `extswitch` remains high, further switchover action is blocked. To select this mode, your two clock sources must be running and the frequency of the two clocks cannot differ by more than 20%. If both clocks are not on the same frequency, but their period difference is within 20%, the clock loss detection block detects the lost clock. The PLL most likely drops out of lock after the PLL clock input switchover and needs time to lock again. |
| **Switchover Delay** (11) | 0–7 | Adds a specific amount of cycle delay to the switchover process. |

***continued...***

(10) This option is only available when selecting **I/O Bank** on the **IOPLL Type**.

(11) This parameter is only available when **Create a second input clk 'refclk1'** is turned on.

| Parameter | Legal Value | Description |
|---|---|---|
| **Access to PLL LVDS_CLK/ LOADEN output port** (10) | **Disabled**, **Enable LVDS_CLK/ LOADEN 0**, or **Enable LVDS_CLK/ LOADEN 0 & 1** | Select **Enable LVDS_CLK/LOADEN 0** or **Enable LVDS_CLK/ LOADEN o & 1** to enable the PLL `lvds_clk` or `loaden` output port.<br>Enables this parameter in case the PLL feeds an LVDS SERDES block with external PLL.<br>When using the I/O PLL `outclk` ports with LVDS ports, `outclk[0..3]` are used for `lvds_clk[0,1]` and `loaden[0,1]` ports, `outclk4` can be used for `coreclk` ports. |
| **Enable access to the PLL DPA output port** (10) | Turn on or Turn off | Turn on to enable the PLL DPA output port. |
| **Enable access to PLL external clock output port** | Turn on or Turn off | Turn on to enable the PLL external clock output port. |
| **Specifies which outclk to be used as extclk_out[0] source** | C0–C6 (I/O bank) | Specifies the `outclk` port to be used as `extclk_out[0]` source. |
| **Specifies which outclk to be used as extclk_out[1] source** | C0–C6 (I/O bank) | Specifies the `outclk` port to be used as `extclk_out[1]` source. |

## 5.3.3. IOPLL IP Core Parameters - Cascading Tab

### Table 9. IOPLL IP Core Parameters - Cascading Tab

| Parameter | Legal Value | Description |
|---|---|---|
| **Connect to an upstream PLL through Core clock Network Cascading (create a permit_cal input signal)** | Turn on or Turn off | Turn on to create an input port to enable destination (downstream) PLL power-up calibration. Connect source (upstream) PLL `locked` signal to this input port. |
| **Create a 'cascade out' signal to connect with a downstream PLL** (12) | Turn on or Turn off | Turn on to create the `cascade_out` port, which indicates that this PLL is a source and connects with a destination (downstream) PLL. |
| **cascade_out source** (12) | 0-6 | Specifies which output clock to be used as cascading source. |
| **Create an adjpllin or cclk signal to connect with an upstream PLL** (12) | Turn on or Turn off | Turn on to create an input port, which indicates that this PLL is a destination and connects with a source (upstream) PLL. |

## 5.3.4. IOPLL IP Core Parameters - Dynamic Reconfiguration Tab

### Table 10. IOPLL IP Core Parameters - Dynamic Reconfiguration Tab for Intel Agilex Devices

| Parameter | Legal Value | Description |
|---|---|---|
| **Enable dynamic reconfiguration of PLL** | Turn on or Turn off | Turn on to enable the dynamic reconfiguration of this PLL (in conjunction with the IOPLL Reconfig Intel FPGA IP core). |
| **Enable access to dynamic phase shift ports** | Turn on or Turn off | Turn on to enable the dynamic phase shift interface with the PLL. |

*continued...*

(12) This option is only available when selecting **I/O Bank** on the **IOPLL Type**.

| Parameter | Legal Value | Description |
|---|---|---|
| **MIF Generation Option** (13) | **Generate New MIF File**, **Add Configuration to Existing MIF File**, or **Create MIF File during IP Generation** | Either create a new `.mif` file containing the current configuration of the I/O PLL by clicking **Create MIF File** or add this configuration to an existing `.mif` file by clicking **Append to MIF File**. A `.mif` file also can be opted to be generated during IP generation.<br>The generated `.mif` file contains current PLL profile and a collection of physical parameters—such as `M`, `N`, `C`, `K`, bandwidth, and charge pump—that defines that PLL. You can use this `.mif` file during dynamic reconfiguration to reconfigure the I/O PLL to its current settings. |
| **Path to New/Existing MIF file** (13) | — | Enter location and file name of the new `.mif` file to be created or existing `.mif` file to be appended. |
| **Name of Current Configuration** (13) | — | Enter the file name of the existing `.mif` file you intend to add to. |

## 5.3.5. IOPLL IP Core Parameters - Advanced Parameters Tab

**Table 11. IOPLL IP Core Parameters - Advanced Parameters Tab for Intel Agilex Devices**

| Parameter | Legal Value | Description |
|---|---|---|
| **Advanced Parameters** | — | Displays a table of physical PLL settings that are implemented based on your input. |

# 5.4. IOPLL IP Core Ports and Signals

**Table 12. IOPLL IP Core Ports for Intel Agilex Devices**

| Port Name | Type | Condition | Description |
|---|---|---|---|
| `refclk` | Input | Required | The reference clock source that drives the I/O PLL. |
| `rst` | Input | Required | The asynchronous reset port for the output clocks. Drive this port high to reset all output clocks to the value of 0. |
| `fbclk` | Input | Optional | The external feedback input port for the I/O PLL.<br>The IOPLL IP core creates this port when the I/O PLL is operating in external feedback mode or zero-delay buffer mode. To complete the feedback loop, a board-level connection must connect the `fbclk` port and the external clock output port of the I/O PLL. |
| `fboutclk` | Output | Optional | The port that feeds the `fbclk` port through the mimic circuitry.<br>The `fboutclk` port is available only if the I/O PLL is in external feedback mode. |
| `zdbfbclk` | Bidirectional | Optional | The bidirectional port that connects to the mimic circuitry. This port must connect to a bidirectional pin that is placed on the positive feedback dedicated output pin of the I/O PLL.<br>The `zdbfbclk` port is available only if the I/O PLL is in zero-delay buffer mode. |
| `locked` | Output | Optional | The IOPLL IP core drives this port high when the PLL acquires lock. The port remains high as long as the I/O PLL is locked. The I/O PLL asserts the `locked` port when the phases and |

***continued...***

(13) This parameter is only available when **Enable dynamic reconfiguration of PLL** is turned on.

| Port Name | Type | Condition | Description |
|---|---|---|---|
| | | | frequencies of the reference clock and feedback clock are the same or within the lock circuit tolerance. When the difference between the two clock signals exceeds the lock circuit tolerance, the I/O PLL loses lock. |
| refclk1 | Input | Optional | Second reference clock source that drives the I/O PLL for clock switchover feature. |
| extswitch | Input | Optional | Active low signal. Assert the extswitch signal low (1'b0) for at least three clock cycles to manually switch the clock. |
| activeclk | Output | Optional | Output signal to indicate which reference clock source is in used by I/O PLL. |
| clkbad | Output | Optional | Output signal that indicates the status of reference clock source is good or bad. |
| cascade_out | Output | Optional | Output signal that feeds into downstream I/O PLL. |
| adjpllin | Input | Optional | Input signal that feeds from upstream I/O PLL. |
| outclk_[] | Output | Optional | Output clock from I/O PLL. |
| permit_cal | Input | Optional | This is an input port for the downstream I/O PLL. Connect this permit_cal port to the locked output port of the upstream I/O PLL. Connecting this permit_cal port ensures that the cascaded I/O PLLs are calibrated in the correct order. |

# 6. IOPLL Reconfig Intel FPGA IP Core

You can use Intel Agilex devices to implement phase-locked loop (PLL) reconfiguration and dynamic phase shift for I/O PLLs.

The Intel Agilex I/O PLL supports dynamic reconfiguration when the device is in user mode. With the dynamic reconfiguration feature, you can reconfigure the I/O PLL settings in real time. You can change the divide settings of the PLL counters and the PLL bandwidth settings (loop filter setting and charge pump setting) through an Avalon® Memory-Mapped (Avalon-MM) interface in the IOPLL Reconfig IP core, without the need to reconfigure the entire FPGA. The Intel Agilex I/O PLL uses divide counters (`N`, `M`, and `C` counters) and a voltage-controlled oscillator (VCO) to synthesize the desired phase and frequency output.

You can use the IOPLL Reconfig IP core as follows:

- Memory Initialization File (`.mif`) streaming reconfiguration
  - Allows the I/O PLL reconfiguration using predefined settings saved in an on-chip ROM. You can store many unique PLL configurations in a single ROM.
  - The `.mif` file is generated automatically by the IOPLL IP core. Using the generated `.mif` file during `.mif` streaming reconfiguration ensures the legality of the new configuration.
  - Intel recommends using this reconfiguration method.
- Advanced mode reconfiguration
  - This method of reconfiguration is for advanced users. You must ensure the reconfigured PLL settings are within the legal range.
  - Enable the **Advanced Reconfiguration** option from the IOPLL Reconfig IP core to reconfigure the individual I/O PLL registers.
  - This method is error prone and may lead to the I/O PLL being reconfigured into an illegal configuration if the reconfiguration is done incorrectly.
- Recalibration of the I/O PLL using `.mif`
  - Perform recalibration of the I/O PLL without any reconfiguration.
  - Trigger recalibration if the reference clock frequency changes.
- I/O PLL clock gating
  - Gate and un-gate I/O PLL output clock 0 to output clock 7 of the I/O PLL.

You can perform dynamic phase shift using the IOPLL Reconfig IP core. The two I/O bank I/O PLLs share the same IOPLL Reconfig IP ports at the I/O bank. Thus, only one of the I/O bank I/O PLLs can be reconfigured in an I/O bank. The fabric-feeding I/O PLL has its own IOPLL Reconfig IP ports and can always be reconfigured.

## 6.1. Release Information for IOPLL Reconfig Intel FPGA IP

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

The IP versioning scheme (X.Y.Z) number changes from one software version to another. A change in:

- X indicates a major revision of the IP. If you update your Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

**Table 13. IOPLL Reconfig Intel FPGA IP Core Current Release Information**

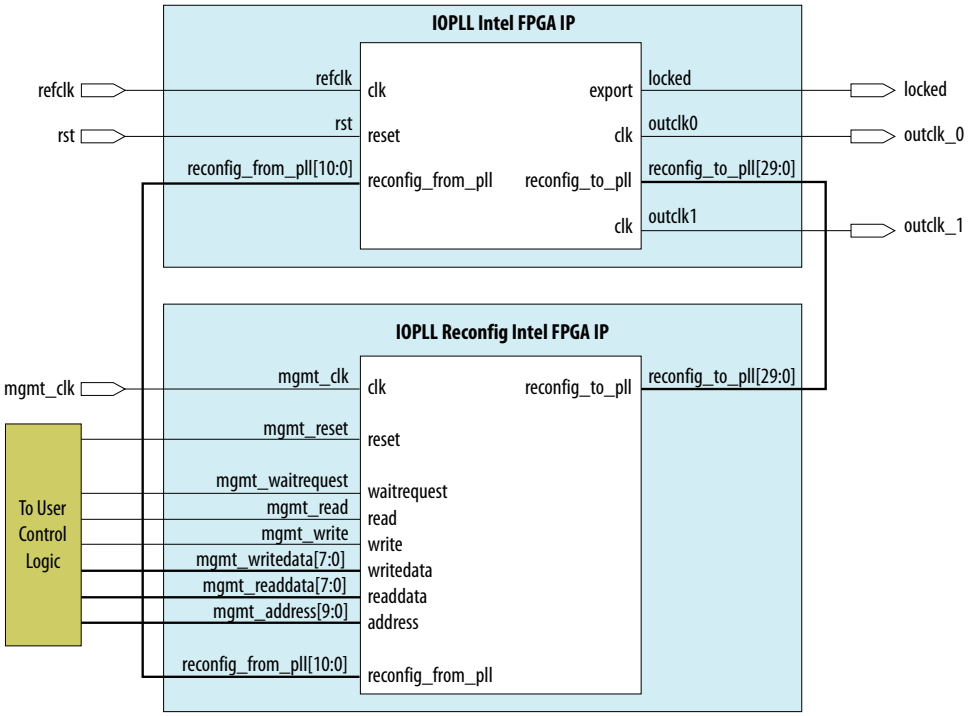
| Item | Description |
|---|---|
| IP Version | 19.3.0 |
| Intel Quartus Prime Version | 19.3 |
| Release Date | 2019.09.30 |

## 6.2. Implementing I/O PLL Reconfiguration in the IOPLL Reconfig IP Core

You can enable the PLL reconfiguration circuitry for the I/O PLL through the Avalon-MM interface in the IOPLL Reconfig IP core.

### 6.2.1. Connectivity between the IOPLL and IOPLL Reconfig IP Cores

**Figure 25. Connectivity between the IOPLL and IOPLL Reconfig IP Cores in the Intel Quartus Prime Software**




### 6.2.2. Connecting the IOPLL and IOPLL Reconfig IP Cores

To connect the IOPLL and IOPLL Reconfig IP cores in your design, follow these steps:

1. Connect the `reconfig_to_pll[29..0]` bus on the IOPLL Reconfig IP core to the `reconfig_to_pll[29..0]` bus on the IOPLL IP core.
2. Connect the `reconfig_from_pll[10..0]` bus on the IOPLL Reconfig IP core to the `reconfig_from_pll[10..0]` bus on the IOPLL IP core.
3. Connect the `mgmt_clk` port to a valid clock source.
4. Connect the `mgmt_reset` port, `mgmt_waitrequest` port, `mgmt_read` port, `mgmt_write` port, `mgmt_readdata[7..0]` bus, `mgmt_writedata[7..0]` bus, and `mgmt_address[9..0]` bus to user control logic to perform read and write operations.

## 6.3. IOPLL Reconfig IP Core Reconfiguration Modes

The IOPLL Reconfig IP core has four functional reconfiguration modes. The reconfiguration operation mode is based on the setting in the `mgmt_address[9:8]` bit.

**Table 14. IOPLL Reconfig IP Core Reconfiguration Modes**

| Reconfiguration Mode | mgmt_address[9:8] |
|---|---|
| .mif streaming reconfiguration | 2'b 00 |
| Advanced mode reconfiguration | 2'b 01 |
| Clock gating reconfiguration | 2'b 10 |
| Dynamic phase shift reconfiguration | 2'b 11 |

After performing dynamic reconfiguration on the I/O PLL that changes the M counter, N counter, bandwidth setting, or charge pump current, the I/O PLL must be recalibrated. For .mif streaming reconfiguration, the recalibration is done automatically. For advanced mode reconfiguration, you must manually trigger the recalibration of the I/O PLL. Recalibration is not needed for clock gating and dynamic phase shift reconfiguration.

## 6.3.1. .mif Streaming Reconfiguration

.mif streaming allows you to dynamically reconfigure the I/O PLL through the IOPLL Reconfig IP core using predefined settings saved in an on-chip RAM. You must generate a .mif file containing these pre-defined configurations, up to 32 I/O PLL configurations, from the IOPLL IP core parameter editor.

To perform .mif streaming reconfiguration, follow these steps:

1. Set mgmt_address[9:8] = 2'b00 to choose the .mif streaming mode and set mgmt_writedata[4:0] to the index of the desired configuration in the .mif file.
2. To start the .mif streaming reconfiguration on the I/O PLL, assert the mgmt_write signal for one mgmt_clk cycle. mgmt_waitrequest is asserted by the IOPLL Reconfig IP core while .mif streaming is in progress.
3. After the reconfiguration is complete, the mgmt_waitrequest signal is de-asserted.
4. In the IOPLL Reconfig IP core parameter editor, select the **Assert waitrequest until IOPLL has locked** option for the I/O PLL to lock. Otherwise, you can wait for the I/O PLL to lock to ensure the I/O PLL reconfiguration is complete.

Recalibration is done automatically for .mif streaming reconfiguration. If you want to manually trigger recalibration using .mif file, follow the steps in the *Recalibration Using .mif* section.

### 6.3.1.1. Recalibration Using .mif

Recalibration using .mif only allows you to recalibrate the I/O PLL but not to reconfigure the I/O PLL. In the IOPLL Reconfig IP core, enable **Recalibration Mode**. When the recalibration is selected, a recalibration.mif file is generated automatically for the recalibration operation.

To perform I/O PLL recalibration using `.mif`, follow these steps:

1. Set `mgmt_address[9:8]` = 2'b00 to choose the `.mif` mode and set `mgmt_writedata[4:0]` = 2'b00.
2. To start the recalibration using `.mif` on the I/O PLL, assert the `mgmt_write` signal for one `mgmt_clk` cycle. `mgmt_waitrequest` is asserted by the IOPLL Reconfig IP core while recalibration is in progress.
3. After the recalibration is complete, the `mgmt_waitrequest` signal is deasserted.

## 6.3.2. Advanced Mode Reconfiguration

In advanced mode, individual I/O PLL setting is reconfigured using the IOPLL Reconfig IP core through the Avalon interface.

Advanced mode reconfiguration is only recommended for advanced users. This reconfiguration mode has several limitations and can cause the I/O PLL to lose lock and can lead to device reliability problems if you set the configuration parameters to the illegal configuration settings. Intel recommends using the `.mif` streaming reconfiguration.

The limitations of using the advanced mode reconfiguration are as follows:

- You must ensure that the configuration setting is a legal value so that the I/O PLL has a legal configuration. To ensure your configuration is legal, refer to the *IOPLL IP Core Parameters - Advanced Parameters Tab* table for the correct configuration settings.
- If the value to be reconfigured makes up only a part of one byte in the I/O PLL's internal memory, you must perform a read-modify-write operation to not overwrite the remaining bits of the byte.
- You must manually trigger recalibration of the I/O PLL after the advanced mode reconfiguration.

***Caution:*** PLL may lose lock and can cause reliability problems to your device if you configure with the wrong PLL setting, configure the wrong bit, or overwrite the whole byte for settings that made up just part of one byte.

To perform I/O PLL reconfiguration using advanced mode, follow these steps:

1. Enable the **Advanced Reconfiguration** option in the IOPLL Reconfig IP core.
2. Set `mgmt_address[9:8]` = 2'b01 to choose the advanced mode reconfiguration.
3. Set the address bus value for `mgmt_address[7:0]` and the data bus value for `mgmt_writedata [7:0]` as the desired PLL setting.

   For more details, refer to the *Address Bus and Data Bus Settings for Advanced Mode Reconfiguration* table.
4. Assert the `mgmt_write` signal for one `mgmt_clk` cycle.
5. Repeat step 1 until step 3 to set address bus and data bus value for the desired I/O PLL reconfiguration setting.
6. After the I/O PLL reconfiguration is complete, you must manually trigger the I/O PLL recalibration.

   For more details about the I/O PLL recalibration, refer to the *Recalibration Using Advanced Mode* section.

#### 6.3.2.1. Recalibration Using Advanced Mode

To perform I/O PLL recalibration using advanced mode, follow these steps:

1. Set `mgmt_address[9:8]` = 2'b01 to choose the advanced mode.
2. Set the `mgmt_writedata[6]` to 1'b1 on `mgmt_address[7:0]` = 8'b01001001 by performing the read-modify-write operation.
3. Set `mgmt_address[7:0]` = 8'b01001010 and `mgmt_wriredata[7:0]` = 8'b00000011 to enable the calibration interface.

### 6.3.3. Clock Gating Reconfiguration

You can gate (disable) and un-gate (enable) I/O PLL output clock 0 to output clock 7 of the I/O PLL. It is easily done by writing one byte to the IOPLL Reconfig IP core, with one bit corresponding to each of the I/O PLL output clocks.

To perform clock gating reconfiguration, follow these steps:

1. Set `mgmt_address[9:8]` to 2'b10 to select clock gating mode and set `mgmt_writedata[7:0]` to indicate desired output clock to be gated.
2. To start the clock gating reconfiguration on the I/O PLL, assert the `mgmt_write` signal for one `mgmt_clk` cycle.
3. The gating changes may not come into effect for multiple clock cycles after `mgmt_waitrequest` has been de-asserted.

### 6.3.4. Dynamic Phase Shift Reconfiguration

The dynamic phase shifts reconfiguration can determine the number of shifts, the direction of the phase shift and the output clock to be shifted.

To perform dynamic phase shift reconfiguration through the IOPLL Reconfig IP core, follow these steps:

1. Set `mgmt_address[9:8]` to 2'b11 to select dynamic phase shift reconfiguration mode.
2. set `mgmt_writedata[7:0]` to indicate the desired number of phase shift, the direction of phase shift, and the desired counter to be shifted.
3. To start the dynamic phase shift reconfiguration on the I/O PLL, assert the `mgmt_write` signal for one `mgmt_clk` cycle. This signal is the equivalent of the `phase_en` signal on the I/O PLL.
4. After the dynamic phase shift is complete, the `mgmt_waitrequest` signal is de-asserted.

## 6.4. Avalon-MM Interface Ports in the IOPLL Reconfig IP Core

**Table 15. Avalon-MM Interface Ports in the IOPLL Reconfig IP Core for Intel Agilex Devices**

| Port | Direction | Description |
|---|---|---|
| mgmt_clk | Input | Dynamic reconfiguration clock that drives the IOPLL Reconfig IP core. The maximum input clock frequency is 100 MHz. This clock can be an independent clock source. It must be free running, which means it cannot be connected to the output of the I/O PLL being reconfigured. |
| mgmt_reset | Input | Active high signal. Synchronous reset input to clear all the data in the IOPLL Reconfig IP core. |
| mgmt_waitrequest | Output | This port goes high when PLL reconfiguration process started and remains high during PLL reconfiguration. After PLL reconfiguration process completed, this port goes low. |
| mgmt_write | Input | Active high signal. Asserts to indicate a write operation. |
| mgmt_read | Input | Active high signal. Asserts to indicate a read operation. |
| mgmt_writedata[7..0] | Input | Writes data to this port when mgmt_write signal is asserted. |
| mgmt_readdata[7..0] | Output | Reads data from this port when mgmt_read signal is asserted. |
| mgmt_address[9..0] | Input | Specifies the address of the data bus for a read or write operation. |
| reconfig_from_pll[10..0] | Input | Bus that connects to reconfig_from_pll[10..0] bus in the IOPLL Intel FPGA IP core. |
| reconfig_to_pll[29..0] | Output | Bus that connects to reconfig_to_pll[29..0] bus in the IOPLL IP core. |

## 6.5. Address Bus and Data Bus Settings

Assign a value of "0" for all the unused bits in the address bus and the data bus during reconfiguration operations.

### 6.5.1. Address Bus and Data Bus Settings for Advanced Mode Reconfiguration

**Table 16. Address Bus and Data Bus Settings for Advanced Mode Reconfiguration**

| Register Name | | Address (Binary) | Counter Bit Setting |
|---|---|---|---|
| M Counter | High Count | 00000100 | • Data[7:0] = high_count<br>• Data[7:0] = low_count<br>• total_count = high_count + low_count |
| | Low Count | 00000111 | |
| | Bypass Enable (14) | 00000101 | • Data[0] = bypass enable<br>— Data[0] = 1, bypass is enabled. The counter is bypassed with counter division value = 1. |

*continued...*

(14) Perform a read-modify-write operation to configure this setting. PLL may lose lock and can cause reliability issue to your device if you configure with the wrong PLL setting, configure the wrong bit, or overwrite the whole byte for settings that made up just part of one byte.

| Register Name | | Address (Binary) | Counter Bit Setting |
|---|---|---|---|
| | Odd Division (14) | 00000110 | • Data[7] = Odd division<br>— Data[7] = 0, odd division is disabled. The selected counter duty cycle = high_count/total_count.<br>— Data[7] = 1, odd division enabled. The selected counter duty cycle = (high_count – 0.5)/total_count. |
| N Counter | High Count | 00000000 | • Data[7:0] = high_count<br>• Data[7:0] = low_count<br>• total_count = high_count + low_count |
| | Low Count | 00000010 | |
| | Bypass Enable (14) | 00000001 | • Data[0] = bypass enable<br>— Data[0] = 1, bypass is enabled. The counter is bypassed with counter division value = 1. |
| | Odd Division (14) | 00000001 | • Data[7] = Odd division<br>— Data[7] = 0, odd division is disabled. The selected counter duty cycle = high_count/total_count.<br>— Data[7] = 1, odd division enabled. The selected counter duty cycle = (high_count – 0.5)/total_count. |
| C0 Counter | High Count | 00011011 | • Data[7:0] = high_count<br>• Data[7:0] = low_count<br>• total_count = high_count + low_count |
| | Low Count | 00011110 | |
| | Bypass Enable (14) | 00011100 | • Data[0] = bypass enable<br>— Data[0] = 1, bypass is enabled. The counter is bypassed with counter division value = 1. |
| | Odd Division (14) | 00011101 | • Data[7] = Odd division<br>— Data[7] = 0, odd division is disabled. The selected counter duty cycle = high_count/total_count.<br>— Data[7] = 1, odd division enabled. The selected counter duty cycle = (high_count – 0.5)/total_count. |
| C1 Counter | High Count | 00011111 | • Data[7:0] = high_count<br>• Data[7:0] = low_count<br>• total_count = high_count + low_count |
| | Low Count | 00100010 | |
| | Bypass Enable (14) | 00100000 | • Data[0] = bypass enable<br>— Data[0] = 1, bypass is enabled. The counter is bypassed with counter division value = 1. |
| | Odd Division (14) | 00100001 | • Data[7] = Odd division<br>— Data[7] = 0, odd division is disabled. The selected counter duty cycle = high_count/total_count.<br>— Data[7] = 1, odd division enabled. The selected counter duty cycle = (high_count – 0.5)/total_count. |
| C2 Counter | High Count | 00100011 | • Data[7:0] = high_count<br>• Data[7:0] = low_count<br>• total_count = high_count + low_count |
| | Low Count | 00100110 | |

*continued...*

| Register Name | | Address (Binary) | Counter Bit Setting |
|---|---|---|---|
| | Bypass Enable [14] | 00100100 | • `Data[0]` = bypass enable<br>— `Data[0]` = 1, bypass is enabled. The counter is bypassed with counter division value = 1. |
| | Odd Division [14] | 00100101 | • `Data[7]` = Odd division<br>— `Data[7]` = 0, odd division is disabled. The selected counter duty cycle = `high_count/total_count`.<br>— `Data[7]` = 1, odd division enabled. The selected counter duty cycle = (`high_count` – 0.5)/`total_count`. |
| C3 Counter | High Count | 00100111 | • `Data[7:0]` = `high_count`<br>• `Data[7:0]` = `low_count`<br>• `total_count` = `high_count` + `low_count` |
| | Low Count | 00101010 | |
| | Bypass Enable [14] | 00101000 | • `Data[0]` = bypass enable<br>— `Data[0]` = 1, bypass is enabled. The counter is bypassed with counter division value = 1. |
| | Odd Division [14] | 00101001 | • `Data[7]` = Odd division<br>— `Data[7]` = 0, odd division is disabled. The selected counter duty cycle = `high_count/total_count`.<br>— `Data[7]` = 1, odd division enabled. The selected counter duty cycle = (`high_count` – 0.5)/`total_count`. |
| C4 Counter | High Count | 00101011 | • `Data[7:0]` = `high_count`<br>• `Data[7:0]` = `low_count`<br>• `total_count` = `high_count` + `low_count` |
| | Low Count | 00101110 | |
| | Bypass Enable [14] | 00101100 | • `Data[0]` = bypass enable<br>— `Data[0]` = 1, bypass is enabled. The counter is bypassed with counter division value = 1. |
| | Odd Division [14] | 00101101 | • `Data[7]` = Odd division<br>— `Data[7]` = 0, odd division is disabled. The selected counter duty cycle = `high_count/total_count`.<br>— `Data[7]` = 1, odd division enabled. The selected counter duty cycle = (`high_count` – 0.5)/`total_count`. |
| C5 Counter | High Count | 00101111 | • `Data[7:0]` = `high_count`<br>• `Data[7:0]` = `low_count`<br>• `total_count` = `high_count` + `low_count` |
| | Low Count | 00110010 | |
| | Bypass Enable [14] | 00110000 | • `Data[0]` = bypass enable<br>— `Data[0]` = 1, bypass is enabled. The counter is bypassed with counter division value = 1. |
| | Odd Division [14] | 00110001 | • `Data[7]` = Odd division<br>— `Data[7]` = 0, odd division is disabled. The selected counter duty cycle = `high_count/total_count`.<br>— `Data[7]` = 1, odd division enabled. The selected counter duty cycle = (`high_count` – 0.5)/`total_count`. |

***continued...***

| Register Name | | Address (Binary) | Counter Bit Setting |
|---|---|---|---|
| C6 Counter | High Count | 00110011 | • Data[7:0] = high_count<br>• Data[7:0] = low_count<br>• total_count = high_count + low_count |
| | Low Count | 00110110 | |
| | Bypass Enable (14) | 00110100 | • Data[0] = bypass enable<br>— Data[0] = 1, bypass is enabled. The counter is bypassed with counter division value = 1.<br>— |
| | Odd Division (14) | 00110101 | • Data[7] = Odd division<br>— Data[7] = 0, odd division is disabled. The selected counter duty cycle = high_count/total_count.<br>— Data[7] = 1, odd division enabled. The selected counter duty cycle = (high_count – 0.5)/total_count. |
| Charge Pump Current (14) | Charge pump setting [2:0] | 00000001 | • Data[6:4] = Charge Pump Setting [2:0]<br>— Configure charge pump setting [2:0] on data bit 4 to 6. |
| | Charge pump setting [5:3] | 00001101 | • Data[7:5] = Charge Pump Setting [5:3]<br>— Configure charge pump setting [5:3] on data bit 5 to 7. |
| Bandwidth Setting (14) | — | 00001010 | • Data[6:3] = Bandwidth Setting<br>— Configure bandwidth setting on data bit 3 to 6. |
| Ripplecap Setting (14) | — | 00001010 | • Data[2:1] = Ripplecap Setting<br>— Configure ripplecap setting on data bit 1 and 2. |
| Calibration (14) | Calibration Request | 01001001 | • Data[6] = Request Calibration<br>— Data[6] = 1, to request calibration |
| | Calibration Enable | 01001010 | • Data[7:0] = Enable Calibration<br>— Data[7:0] = 8'b00000011, to enable calibration |

### 6.5.1.1. Data Bus Setting for Bandwidth Control and Charge Pump

**Table 17. I/O Bank I/O PLL Data Bus Setting for Bandwidth Control and Charge Pump (For Low Bandwidth)**

| Multiply Factor (15) | Low Bandwidth | | |
|---|---|---|---|
| | Bandwidth Control Setting Data [6:3] | Charge Pump Setting [2:0] Address = 00000001 Data [6:4] | Charge Pump Setting [5:3] Address = 00001101 Data [7:5] |
| 4–5 | 4'b0011 | 3'b011 | 3'b000 |
| 6–7 | 4'b0011 | 3'b011 | 3'b000 |
| 8–10 | 4'b0011 | 3'b100 | 3'b000 |

*continued...*

(15) If you select the **Use Nondedicated Feedback Path** option under the **Normal** or **Source Synchronous** compensation mode, the Multiply Factor is M x $C_i$, where M is M counter value while $C_i$ is the Compensated Outclk C counter value. Else, the Multiply Factor is only the M counter value.

| Multiply Factor (15) | Low Bandwidth | | |
|---|---|---|---|
| | Bandwidth Control Setting Data [6:3] | Charge Pump Setting [2:0] Address = 00000001 Data [6:4] | Charge Pump Setting [5:3] Address = 00001101 Data [7:5] |
| 11–15 | 4'b0100 | 3'b100 | 3'b000 |
| 16–20 | 4'b0101 | 3'b100 | 3'b000 |
| 21–23 | 4'b0101 | 3'b101 | 3'b000 |
| 24–43 | 4'b0110 | 3'b100 | 3'b000 |
| 44–64 | 4'b0111 | 3'b100 | 3'b000 |
| 65–85 | 4'b1000 | 3'b100 | 3'b000 |
| 86–124 | 4'b1010 | 3'b011 | 3'b000 |
| 125–160 | 4'b1010 | 3'b011 | 3'b000 |

**Table 18. I/O Bank I/O PLL Data Bus Setting for Bandwidth Control and Charge Pump (For Medium Bandwidth)**

| Multiply Factor (15) | Medium Bandwidth | | |
|---|---|---|---|
| | Bandwidth Control Setting Data [6:3] | Charge Pump Setting [2:0] Address = 00000001 Data [6:4] | Charge Pump Setting [5:3] Address = 00001101 Data [7:5] |
| 4–5 | 4'b0010 | 3'b110 | 3'b000 |
| 6–7 | 4'b0011 | 3'b101 | 3'b000 |
| 8–10 | 4'b0011 | 3'b101 | 3'b000 |
| 11–15 | 4'b0011 | 3'b110 | 3'b000 |
| 16–20 | 4'b0100 | 3'b110 | 3'b000 |
| 21–23 | 4'b0100 | 3'b111 | 3'b000 |
| 24–43 | 4'b0101 | 3'b110 | 3'b000 |
| 44–64 | 4'b0110 | 3'b110 | 3'b000 |
| 65–85 | 4'b0111 | 3'b110 | 3'b000 |
| 86–124 | 4'b1000 | 3'b110 | 3'b000 |
| 125–160 | 4'b1010 | 3'b100 | 3'b000 |

(15) If you select the **Use Nondedicated Feedback Path** option under the **Normal** or **Source Synchronous** compensation mode, the Multiply Factor is M x $C_i$, where M is M counter value while $C_i$ is the Compensated Outclk C counter value. Else, the Multiply Factor is only the M counter value.

**Table 19. I/O Bank I/O PLL Data Bus Setting for Bandwidth Control and Charge Pump (For High Bandwidth)**

| Multiply Factor (15) | High Bandwidth | | |
|---|---|---|---|
| | Bandwidth Control Setting Data [6:3] | Charge Pump Setting [2:0] Address = 00000001 Data [6:4] | Charge Pump Setting [5:3] Address = 00001101 Data [7:5] |
| 4–5 | 4'b0010 | 3'b001 | 3'b001 |
| 6–7 | 4'b0010 | 3'b010 | 3'b001 |
| 8–10 | 4'b0010 | 3'b011 | 3'b001 |
| 11–15 | 4'b0011 | 3'b001 | 3'b001 |
| 16–20 | 4'b0011 | 3'b010 | 3'b001 |
| 21–23 | 4'b0100 | 3'b010 | 3'b001 |
| 24–43 | 4'b0100 | 3'b011 | 3'b001 |
| 44–64 | 4'b0101 | 3'b011 | 3'b001 |
| 65–85 | 4'b0101 | 3'b011 | 3'b001 |
| 86–124 | 4'b1001 | 3'b110 | 3'b000 |
| 125–160 | 4'b1001 | 3'b110 | 3'b000 |

**Table 20. Fabric-Feeding I/O PLL Data Bus Setting for Bandwidth Control and Charge Pump (For Medium Bandwidth)**

| Multiply Factor (15) | Medium Bandwidth | | |
|---|---|---|---|
| | Bandwidth Control Setting Data [6:3] | Charge Pump Setting [2:0] Address = 00000001 Data [6:4] | Charge Pump Setting [5:3] Address = 00001101 Data [7:5] |
| 4–5 | 4'b0011 | 3'b100 | 3'b000 |
| 6–7 | 4'b0100 | 3'b100 | 3'b000 |
| 8–10 | 4'b0101 | 3'b100 | 3'b000 |
| 11–15 | 4'b0110 | 3'b100 | 3'b000 |
| 16–20 | 4'b0111 | 3'b100 | 3'b000 |
| 21–23 | 4'b0111 | 3'b100 | 3'b000 |
| 24–43 | 4'b1001 | 3'b011 | 3'b000 |
| 44–64 | 4'b1010 | 3'b011 | 3'b000 |
| 65–85 | 4'b1010 | 3'b100 | 3'b000 |
| 86–104 | 4'b1010 | 3'b101 | 3'b000 |
| 105–140 | 4'b1011 | 3'b100 | 3'b000 |
| 141–160 | 4'b1011 | 3'b100 | 3'b000 |

**Table 21. Fabric-Feeding I/O PLL Data Bus Setting for Bandwidth Control and Charge Pump (For High Bandwidth)**

| Multiply Factor (15) | High Bandwidth | | |
|---|---|---|---|
| | Bandwidth Control Setting Data [6:3] | Charge Pump Setting [2:0] Address = 00000001 Data [6:4] | Charge Pump Setting [5:3] Address = 00001101 Data [7:5] |
| 4–5 | 4'b0010 | 3'b001 | 3'b001 |
| 6–7 | 4'b0011 | 3'b000 | 3'b001 |
| 8–10 | 4'b0011 | 3'b000 | 3'b001 |
| 11–15 | 4'b0100 | 3'b000 | 3'b001 |
| 16–20 | 4'b0110 | 3'b110 | 3'b000 |
| 21–23 | 4'b0111 | 3'b101 | 3'b000 |
| 24–43 | 4'b1000 | 3'b101 | 3'b000 |
| 44–64 | 4'b1001 | 3'b101 | 3'b000 |
| 65–85 | 4'b1001 | 3'b110 | 3'b000 |
| 86–104 | 4'b1001 | 3'b110 | 3'b000 |
| 105–140 | 4'b1010 | 3'b101 | 3'b000 |
| 141–160 | 4'b1010 | 3'b100 | 3'b000 |

### 6.5.1.2. Data Bus Setting for Ripplecap

**Table 22. I/O Bank I/O PLL Data Bus Setting for Ripplecap**

| Multiply Factor (16) | Ripplecap Setting Address = 00001010 Data [2:1] | | |
|---|---|---|---|
| | Low Bandwidth | Medium Bandwidth | High Bandwidth |
| 4–5 | 2'b01 | 2'b01 | 2'b00 |
| 6–7 | 2'b11 | 2'b01 | 2'b01 |
| 8–10 | 2'b11 | 2'b01 | 2'b01 |
| 11–15 | 2'b11 | 2'b01 | 2'b01 |
| 16–20 | 2'b11 | 2'b01 | 2'b01 |
| 21–23 | 2'b11 | 2'b01 | 2'b01 |
| 24–43 | 2'b11 | 2'b01 | 2'b01 |
| 44–64 | 2'b11 | 2'b01 | 2'b01 |

*continued...*

(16) If you select the **Use Nondedicated Feedback Path** option under the **Normal** or **Source Synchronous** compensation mode, the Multiply Factor is M x $C_i$, where M is M counter value while $C_i$ is the Compensated Outclk C counter value. Else, the Multiply Factor is only the M counter value.

| Multiply Factor (16) | Ripplecap Setting Address = 00001010 Data [2:1] | | |
|---|---|---|---|
| | Low Bandwidth | Medium Bandwidth | High Bandwidth |
| 65–85 | 2'b11 | 2'b01 | 2'b01 |
| 86–124 | 2'b11 | 2'b01 | 2'b01 |
| 125–160 | 2'b11 | 2'b01 | 2'b01 |

**Table 23. Fabric-Feeding I/O PLL Data Bus Setting for Ripplecap**

| Multiply Factor (16) | Ripplecap Setting Address = 00001010 Data [2:1] | |
|---|---|---|
| | Medium Bandwidth | High Bandwidth |
| 4–5 | 2'b01 | 2'b00 |
| 6–7 | 2'b01 | 2'b00 |
| 8–10 | 2'b01 | 2'b01 |
| 11–15 | 2'b01 | 2'b01 |
| 16–20 | 2'b01 | 2'b01 |
| 21–23 | 2'b01 | 2'b01 |
| 24–43 | 2'b01 | 2'b01 |
| 44–64 | 2'b01 | 2'b01 |
| 65–85 | 2'b01 | 2'b01 |
| 86–104 | 2'b01 | 2'b01 |
| 105–140 | 2'b01 | 2'b01 |
| 141–160 | 2'b01 | 2'b00 |

## 6.5.2. Output Clock and the Corresponding Data Bit Setting for Clock Gating Reconfiguration

**Table 24. Output Clock and the Corresponding Data Bit Setting for Clock Gating Reconfiguration**

| Output Clock | Data Bus Bit Setting (Binary) | |
|---|---|---|
| C0 | data[0] | Gated = 1'b0<br>Ungated = 1'b1 |
| C1 | data[1] | |
| C2 | data[2] | |
| C3 | data[3] | |

*continued...*

(16) If you select the **Use Nondedicated Feedback Path** option under the **Normal** or **Source Synchronous** compensation mode, the Multiply Factor is M x $C_i$, where M is M counter value while $C_i$ is the Compensated Outclk C counter value. Else, the Multiply Factor is only the M counter value.

| Output Clock | Data Bus Bit Setting (Binary) | |
|---|---|---|
| C4 | data[4] | |
| C5 | data[5] | |
| C6 | data[6] | |

## 6.5.3. Data Bus Setting for Dynamic Phase Shift for IOPLL Reconfig IP Core

**Table 25. Data Bus Setting for Dynamic Phase Shift for IOPLL Reconfig IP Core**

| Write Data Bus Setting | Description | |
|---|---|---|
| data[2:0] | Determines the number of phase shifts per dynamic phase shift operation. Up to seven phase shifts per operation are possible. Each phase shift step is equal to 1/8 of I/O PLL VCO period. | |
| data[3] | Determines the direction of dynamic phase shift. When data[3] = 0, phase shift is in negative direction. When data[3] = 1, phase shift is in positive direction. | |
| data[7:4] | Determines the counter to be selected to perform dynamic phase shift operation. | |
| | **Counter Name** | **data[7:4]** |
| | C0 | 4'b0000 |
| | C1 | 4'b0001 |
| | C2 | 4'b0010 |
| | C3 | 4'b0011 |
| | C4 | 4'b0100 |
| | C5 | 4'b0101 |
| | C6 | 4'b0110 |
| | All C counters | 4'b1111 |

# 7. Document Revision History for the Intel Agilex Clocking and PLL User Guide

| Document Version | Intel Quartus Prime Version | Changes |
|---|---|---|
| 2019.12.18 | 19.3 | Removed `scanclk` signal in the *Guideline: I/O PLL Reconfiguration* section. |
| 2019.10.31 | 19.3 | • Added the *Examples of Clock Networks Sizes Using Intel Agilex Programmable Clock Routing* diagram in the *Programmable Clock Routing* section.<br>• Updated the number of resources available in the *Programmable Clock Routing Resources for Intel Agilex Devices* table.<br>• Updated the *PLL Features in Intel Agilex Devices* table.<br>— Updated C counter divide factor range from '1 to 510' to '1 to 512'.<br>— Added a note to dedicated external clock outputs.<br>— Removed the following PLL features for fabric-feeding I/O PLL.<br>• Dedicated external clock outputs<br>• External feedback input pin<br>• External feedback compensation<br>— Added spread-spectrum input clock tracking feature.<br>• Updated the *PLL Usage* section.<br>• Removed external feedback mode in the *Fabric-Feeding I/O PLL High-Level Block Diagram for Intel Agilex Devices* diagram.<br>• Clarified that EFB mode is only supported for I/O bank I/O PLL.<br>• Updated the *PLL Cascading* section.<br>— Added description that Intel Agilex devices does not support I/O PLL cascading within the same I/O bank.<br>— Updated `outclk[8:0]` to `outclk[6:0]` in the I/O-PLL-to-I/O-PLL cascading diagrams.<br>• Added the following guidelines:<br>— Guideline: I/O PLL Reconfiguration<br>— Clocking Constraints<br>— IP Core Constraints<br>• Added information for the following IP cores:<br>— Clock Control Intel FPGA IP version 1.0.0<br>— IOPLL Intel FPGA IP version 19.3.0<br>— IOPLL Reconfig Intel FPGA IP version 19.3.0 |
| 2019.04.02 | — | Initial release. |