ALTPLL (Phase-Locked Loop) IP Core User Guide

2014.08.18

ug-altpll

Subscribe Send Feedback

The Altera Phase-Locked Loop (ALTPLL) IP core implements phase lock loop (PLL) circuitry. A PLL is a feedback control system that automatically adjusts the phase of a locally generated signal to match the phase of an input signal. PLLs operate by producing an oscillator frequency to match the frequency of an input signal. In this locked condition, any slight change in the input signal first appears as a change in phase between the input signal and the oscillator frequency.

This phase shift then acts as an error signal to change the frequency of the local PLL oscillator to match the input signal. The locking-onto-a-phase relationship between the input signal and the local oscillator accounts for the name phase-locked loop. PLLs are often used in high-speed communication applications

You can use the Quartus® II IP Catalog and parameter editor to specify PLL parameters .

Note: This IP core is not supported for Arria 10 designs.

Related Information

- **Introduction to Altera IP Cores**
- **Altera IP Release Notes**

ALTPLL Features

The PLL types, operation modes, and advanced features are available for configuration in the ALTPLL IP core. Each PLL feature includes a table that compares the PLL feature in the supported devices, and describes the relevant parameter settings.

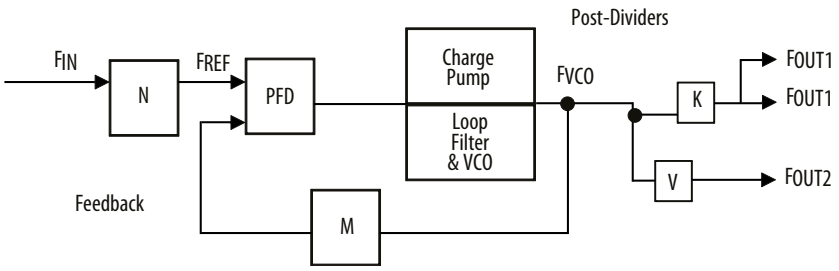
Phase-Locked Loop

The Phase-Locked Loop (PLL) is a closed-loop frequency-control system that compares the phase difference between the input signal and the output signal of a voltage-controlled oscillator (VCO). The negative feedback loop of the system forces the PLL to be phase-locked.

PLLs are widely used in telecommunications, computers, and other electronic applications. You can use the PLL to generate stable frequencies, recover signals from a noisy communication channel, or distribute clock signals throughout your design.

Building Blocks of a PLL

Figure 1: PLL Block Diagram



The PLL consists of a pre-divider counter (*N* counter), a phase-frequency detector (PFD) circuit, a charge pump, loop filter, a VCO, a feedback multiplier counter (*M* counter), and post-divider counters (*K* and *V* counters).

The PFD detects the differences in phase and frequency between its reference signal (f_{REF}) and feedback signal (Feedback), controls the charge pump, and controls a loop filter that converts the phase difference to a control voltage. This voltage controls the VCO.

Based on the control voltage, the VCO oscillates at a higher or lower frequency, which affects the phase and frequency of the Feedback signal. After the f_{REF} signal and the Feedback signal have the same phase and frequency, the PLL is said to be phase-locked.

Inserting the *M* counter in the feedback path causes the VCO to oscillate at a frequency that is *M* times the frequency of the f_{REF} signal. The f_{REF} signal is equal to the input clock (f_{IN}) divided by the pre-scale counter (*N*).

The reference frequency is described by the equation $f_{REF} = f_{IN}/N$. The VCO output frequency is $f_{VCO} = f_{IN} \times M/N$, and the output frequency of the PLL is described by the equation $f_{OUT} = (f_{IN} \times M)/(N \times K)$ for the signals.

PLL Behavior

- **PLL lock time**—Also known as the PLL acquisition time, PLL lock time is the amount of time required by the PLL to attain the target frequency and phase relationship after power-up, after a programmed output frequency change, or after a reset of the PLL. Simulation software does not model a realistic PLL lock time. Simulation shows an unrealistically fast lock time.
- **PLL resolution**—The minimum frequency increment value of a PLL VCO. The value is based on the number of bits in the *M* and *N* counter.
- **PLL sample rate**—The f_{REF} sampling frequency required to perform the phase and frequency correction in the PLL. The PLL sample rate is f_{REF}/N.

Types of PLLs

The types of PLL supported by the IP core depend on the device family. Device families typically support one or two PLL types. For example, the Stratix series supports two types of PLLs, and the Cyclone series supports only one type. The two PLL types supported within a device family are identical in their analog portions and differ slightly in the digital portion, for example, more counters on one type than another.

Parameter Setting

You select the PLL type on the **General/Modes** page of the ALTPLL parameter editor. The list of available PLL types to choose from depends on the selected device family. If you select **Select the PLL type automatically**, the ALTPLL parameter editor selects the best possible PLL type, based on other options that you set in the ALTPLL parameter editor.

Related Information

- **Determining the PLL Lock Range** on page 6
- **Expanding the PLL Lock Range** on page 6
- **Output Clocks** on page 9
- **Ports and Parameters** on page 38

Total Number of PLL Available in Each Supported Device Family

The following table lists the total number of PLLs available for configuration and the PLL types supported by the ALTPLL IP core for each device family.

Table 1: Total Number of PLLs per Device Family

Device Family	Total Number of PLLs	PLL Types
Arria GX	8	Enhanced and Fast
Arria II GX	6	Left_Right
Stratix IV	12	Top_Bottom and Left_Right
Stratix III	12	Top_Bottom and Left_Right
Stratix II	12	Enhanced and Fast
Stratix II GX	8	Enhanced and Fast
Stratix	12	Enhanced and Fast
Stratix GX	8	Enhanced and Fast
Cyclone IV	4	Cyclone IV PLL
Cyclone III	4	Cyclone III PLL
Cyclone II	4	Cyclone II PLL
Cyclone	2	Cyclone PLL

Operation Modes

The ALTPLL IP core supports up to five different clock feedback modes, depending on the selected device family. Each mode allows clock multiplication and division, phase shifting, and duty-cycle programming.

The following list describes the operation modes for the ALTPLL IP core:

- **Normal mode**—The PLL feedback path source is a global or regional clock network, minimizing clock delay to registers for that clock type and specific PLL output. You can specify PLL output that is compensated in normal mode.
- **Source-Synchronous mode**—The data and clock signals arrive at the same time at the data and clock input pins. In this mode, the signals are guaranteed to have the same phase relationship at the clock and data ports of any Input Output Enable register.
- **Zero-Delay Buffer mode**—The PLL feedback path is confined to the dedicated PLL external clock output pin. The clock port driven off-chip is phase aligned with the clock input for a minimal delay between the clock input and the external clock output.
- **No Compensation mode**—The PLL feedback path is confined to the PLL loop. It has no clock network or other external source. A PLL in no-compensation mode has no clock network compensation, but clock jitter is minimized.
- **External Feedback mode**—The PLL compensates for the `fbin` feedback input to the PLL. The delay between the input clock pin and the feedback clock pin is minimized.

Operation Modes Supported in Each Device Family

The following table summarizes the operation modes supported for each device family.

Table 2: PLL Types and Modes Supported in Different Device Families

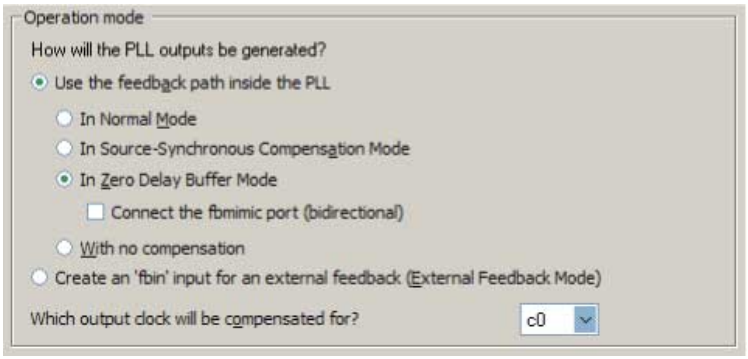
Device Family	Normal	Source-Synchronous	Zero-Delay Buffer	No Compensation	External Feedback
Arria GX	All PLL types	All PLL types	Enhanced PLL	All PLL types	Enhanced PLL
Arria II GX	Left_Right PLL	Left_Right PLL	Left_Right PLL	Left_Right PLL	—
Stratix IV	All PLL types	All PLL types	All PLL types	All PLL types	All PLL types
Stratix III	All PLL types	All PLL types	Enhanced PLL	All PLL types	Enhanced PLL
Stratix II	All PLL types	All PLL types	Enhanced PLL	All PLL types	Enhanced PLL
Stratix II GX	All PLL types	All PLL types	Enhanced PLL	All PLL types	Enhanced PLL
Stratix	All PLL types	—	Enhanced PLL	All PLL types	Enhanced PLL
Stratix GX	All PLL types	—	Enhanced PLL	All PLL types	Enhanced PLL
Cyclone IV	All PLL types	All PLL types	All PLL types	All PLL types	—
Cyclone III	All PLL types	All PLL types	All PLL types	All PLL types	—
Cyclone II	All PLL types	All PLL types	All PLL types	All PLL types	—
Cyclone	All PLL types	—	All PLL types	All PLL types	—

Parameter Settings

Describes how to set the operation mode for the PLL using the ALTPLL parameter editor. The parameter settings are located on the **General/Modes** page of the ALTPLL parameter editor.

The following figure shows the options you can select from the page.

Figure 2: Operation Mode Options



The following table lists the options you can select from the page.

Table 3: Operation Mode Options and Descriptions

Option	Description
Use the feedback path inside the PLL	Specify which operation mode to use. For source-synchronous mode, zero-delay buffer mode, and external feedback mode, you must make PLL Compensation assignments using the Assignment Editor in addition to setting the appropriate mode in the IP core. The assignment allows you to specify an output pin as a compensation target for a PLL in zero-delay buffer mode or external feedback mode, or to specify an input pin or group of input pins as compensation targets for a PLL in source-synchronous mode.
Create an 'fbin' input for an external feedback (External Feedback Mode)	Select this option to set the PLL in external feedback mode. The `fbin` port is the input port to the PLL from the external feedback path. In this mode, the PLL compensates for the `fbin` port. The delay between the input clock pin and the feedback clock pin is minimized.
Which output clock will be compensated for?	Specify which output port of the PLL is to be compensated for. The drop down list contains all output clock ports for the selected device. The correct output clock selection depends on the operation mode that you select. For example, for normal mode, select the core output clock. For zero-delay buffer mode or external feedback mode, select the external output clock.

The following figure shows the options you can select from the page.

Figure 3: General Options

The following table lists the options you can select from the page.

Table 4: Operation Mode Options and Descriptions

Option	Description
Which device speed grade will you be using?	Specify the speed grade if you are not already using a device with the fastest speed. The lower the number, the faster the speed grade.
What is the frequency of the inclock0 input?	Specify the frequency of the input clock signal.
Set up PLL in LVDS mode	Select this option when you want the PLL to supply the necessary clocking signals to the LVDS transmitter or receiver. In this mode, the PLL type and operation mode are forced to fast PLL and normal mode, respectively. This option creates two new output ports —`sclkout` and `enable`. This option is available only for the Arria GX, Stratix II, Stratix II GX, andHardCopy II device families.
Data rate	Specify the data rate for the PLL in LVDS mode. This option is available only if **Set up PLL in LVDS mode** is enabled.

Determining the PLL Lock Range

The PLL lock range is the range between the minimum (Freq min lock parameter) and maximum (Freq min lock parameter) input frequency values for which the PLL can achieve lock. The Quartus II software shows these input frequency values in the PLL Summary report located under Resource Section of the Fitter folder in the Compilation Report. Changing the input frequency may cause the PLL to lose lock, but while the input clock remains within the minimum and maximum frequency specifications, the PLL is able to achieve lock.

Expanding the PLL Lock Range

The Quartus II software does not necessarily pick values for the PLL parameters to maximize the lock range. For example, you specify a 75 MHz input clock in the ALTPLL parameter editor, the actual PLL lock range may be between 70 MHz to 90 MHz. If your application requires a lock range of 50 MHz to 100 MHz, the default lock range of this PLL is insufficient.

For devices that support clock switchover in PLLs, you can use the ALTPLL parameter editor to maximize the lock range.

To extract valid parameter values to maximize your PLL lock range, perform the following steps:

1. In the schematic editor, double-click the ALTPLL instance in your design to open the ALTPLL parameter editor.
2. For **What is the frequency of your inclock0 input?**, type the value of the low end of your desired PLL lock range. For example, if your application requires a lock range of 50 MHz to 100 MHz, type `50` MHz.
3. Turn on **Create output file(s) using the 'Advanced' PLL parameters**.
4. Turn on **Create an 'inclk1' for a second inclk** and enter the high end of your lock range as the frequency for `inclk1`. For example, if your application requires a lock range of 50 MHz to 100 MHz, type `100` MHz.
5. Complete the remaining pages in the ALTPLL parameter editor.
6. Compile your project and note the lock range shown in the PLL Summary report. If it is satisfactory, note all of the values for the PLL from this report, such as the *M* value, *N* value, charge pump current, loop filter resistance, and loop filter capacitance.
7. In the schematic editor, double-click the ALTPLL instance in your design to open the ALTPLL parameter editor.
8. Turn off **Create an 'inclk1' for a second inclk**.
9. Click **Finish** to update the PLL wrapper file.
10. In a text editor, open the PLL wrapper file. If the wrapper file is in Verilog format, go to the **defparam** section. If the wrapper file is in VHDL HDL, go to the generic map section. Modify all of the values for the parameters listed in step 6. Save the changes.
11. Compile your project.
12. Check the PLL Summary report to confirm the PLL lock range meets your requirements. The modified PLL should have the desired lock range.

If your input clock frequency is too close to the end of the desired PLL lock range—for example the low end of the desired lock range is 50 MHz and the input clock frequency is 50 MHz, the PLL might not maintain lock when the input clock has jitter or the frequency drifts below 50 MHz. You may choose to expand your PLL lock range to ensure your expected input clock frequency is further from the end of the range. For this example, you can enter 45 MHz and 105 MHz to ensure that your target lock range of 50 MHz to 100 MHz is within the PLL lock range.

The Quartus II software prompts an error message if it is unable to implement your preferred lock range using this procedure. Therefore, you have to look into other options, such as PLL reconfiguration to support your input frequency range.

Setting Up Stratix III and Stratix IV PLLs for LVDS Interfacing

The ALTLVDS IP core provides SERDES transmitter and receiver functionality commonly used in LVDS interfacing.

The following table lists the options and values to configure a PLL on a Stratix III or Stratix IV device to clock an ALTLVDS IP core.

Table 5: Options to Configure a PLL on a Stratix III or Stratix IV Device

Option	Value
Which PLL type will you be using?	Left_Right PLL
How will the PLL outputs be generated?	In Source-Synchronous Compensation mode

Option	Value		
On the Output clocks page	c0 This clock signal is the high-speed serial clock (fast clock) signal connected to the rx_inclock or tx_inclock port of the ALTLVDS IP core. Output frequency = data rate Phase shift = -180 degrees Duty cycle = 50%	c1 This clock signal is the load enable signal connected to the rx_enable or tx_enable port of the ALTLVDS IP core. Output frequency = data rate/deserialization factor Phase shift = [(deserialization factor – 2)/deserialization factor] × 360 degrees Duty cycle = (100/deserialization factor) %	c2 This clock signal is the slow clock signal that feeds the synchronization register of the ALTLVDS IP core. Output frequency = data rate/deserialization factor Phase shift = (-180/deserialization factor) degrees Duty cycle = 50%

Related Information

SERDES Transmitter/Receiver (ALTLVDS) IP Core User Guide

Simulating External Feedback Board Delay

The PLL external feedback board delay option is available for Arria GX, Cyclone, HardCopy series, Stratix, Stratix GX, Stratix II, and Stratix II GX device families only.

The functional and timing models of these devices do not support the simulation of external feedback. To simulate the external feedback mode, perform the following steps:

1. In the Quartus II software, open an existing project or create a new project.
2. On the Assignments menu, click **Assignment Editor**.
3. In the **Category** bar, under Timing, click **All**.
4. In the spreadsheet, double-click an empty row in the **To** cell and either type in the pin name or click the arrow to use the **Node Finder** to search for the external feedback input pin.
5. Double-click the **Assignment Name** cell, and select **PLL External Feedback Board Delay**.
6. In the **Value** cell, double-click and type the amount of time for the signal to propagate between the external clock output pin through the trace on the board and into the external feedback input pin.
7. Simulate your design.

The behavioral models for the ALTPLL IP core reside in the **\quartus\eda\sim_lib** directory. The **altera_mf.vhd** file contains the VHDL behavioral models and the **altera_mf.v** file contains the Verilog HDL behavioral models. The behavioral model does not perform parameter error checking, so you must specify valid values.

Output Clocks

The PLL can generate a number of clock output signals depending on the PLL type and the device family that you select in the ALTPLL parameter editor. For example, in a Stratix IV device, a Left_Right PLL can generate seven clock output signals, and a Top_Bottom PLL can generate as many as 10 clock output signals. The generated clock output signals are used to clock the core or external blocks outside of the core.

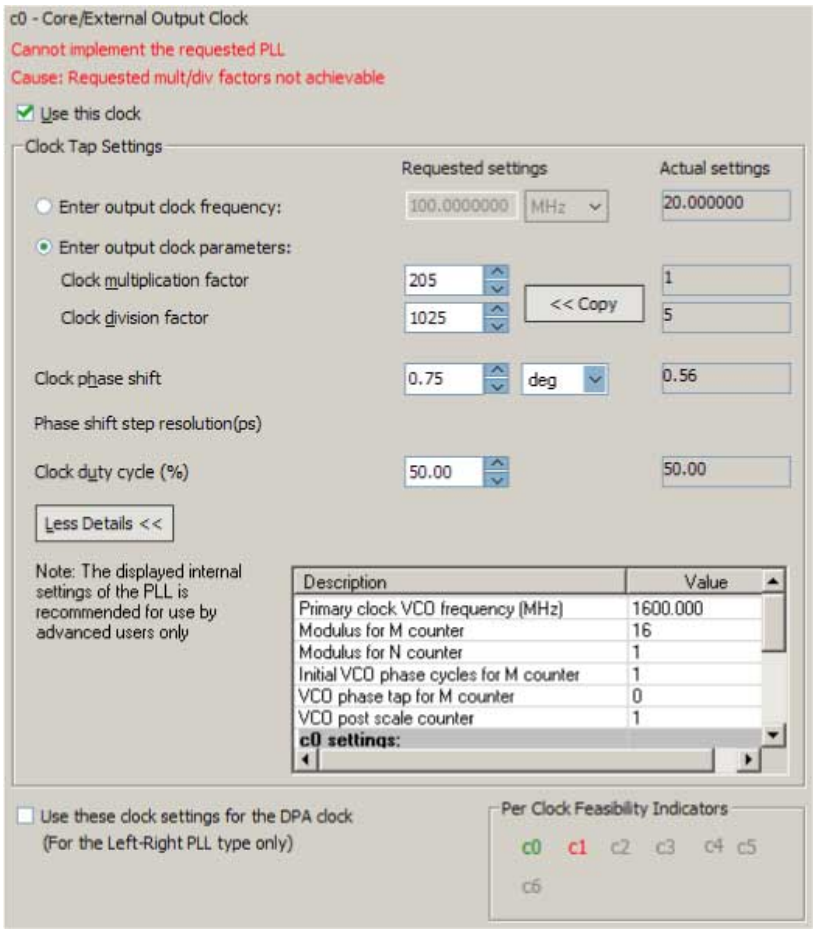
The ALTPLL IP core does not have a dedicated output enable port, you can disable the PLL output. You can use the `pllena` signal or the `areset` signal to disable the PLL output counters, and thereby disable the PLL output clocks. Another possible method is to feed the PLL output clock signals to the ALTIOBUF IP core and use the enable output ports of the resulting buffers to disable the signals.

Parameter Settings

The **Output Clocks** page of the ALTPLL parameter editor contains the parameter settings for the clock output signals. The output clock port can be used as a core output clock or an external output clock port. The core output clock is used to feed the FPGA core and the external output clock is used to feed the dedicated pins on the FPGA.

The following figure shows a screenshot of the page to configure the `c0` clock output signal of the ALTPLL IP core.

Figure 4: Output Clocks



Each option has the following two columns:

- Requested settings
- Actual settings

The requested settings are the settings that you want to implement, and the actual settings are the settings closest values that can be implemented in the PLL circuit to best approximate the requested settings. Use the values in the actual settings column as a guide to adjust the requested settings. If the requested settings for one of the output clocks cannot be approximated, the ALTPLL parameter editor produces a warning message at the top of every page. To determine the output clocks that contain unachievable settings, turn on **Per Clock Feasibility Indicators** at the bottom of the ALTPLL parameter editor. The output clock name in red is the name of the clock with unachievable settings. The clock listed in green has no settings issues, and the grayed-out names are the unselected output clocks. You must adjust the requested settings for the affected output clocks to resolve the warning messages, or use another Altera device that meets your desired timing specifications.

To generate an output clock port in your ALTPLL IP core variation, select **Use this clock**.The output clock port that is to be compensated for is enabled by default. It cannot be disabled, unless you select a different output clock port to be compensated for.

The rest of the options on the page allow you to set the following output clock values:

- frequency
- phase shift
- duty cycle

The phase shift option allows you to set the programmable phase shift for an output clock signals. The smallest phase shift is 1/8th of VCO period. For degree increments, the maximum step size is 45 degrees. You can set smaller steps using the clock multiplication and division factors options. For example, if the post-scale counter is 32, the smallest phase shift step is 0.1°. The up and down buttons let you cycle through phase shift values. Alternatively, you can enter a number in the phase shift field manually instead of using the buttons.

Instead of specifying frequency of the output clock signal, you can also specify the multiplication and division factors of the signal in the requested settings column.

The following figure shows the formula for an output clock frequency.

Figure 5: PLL Output Clock Frequency

$$\text{Output clock frequency} = \text{Input clock frequency} \times \frac{\text{multiplication factor}}{\text{division factor}}$$

The ALTPLL parameter editor calculates the simplest fraction, and displays it in the actual settings column. For example, if the input clock frequency is 100 MHz, and the requested multiplication and division factors are 205 and 1025 respectively, the output clock frequency is calculated as 100 × 205/1025=20 MHz. The actual settings reflect the simplest fraction — the actual multiplication factor is 1, and the actual division factor is 5. You can use the copy button to copy values from the actual settings to the requested settings.

The actual values of multiplication and division factors are affected when you select **Use these clock settings for the DPA clock** or **Set up PLL in LVDS mode**.

Select **Use these clock settings for the DPA clock** if you want the output clock signal of the PLL to drive the input clock port of the DPA block in the ALTLVDS IP core. This option is available only for Stratix III and Stratix IV devices.

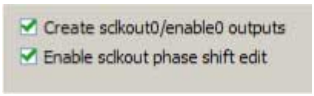
Parameter Settings When "Set up PLL in LVDS mode" Option is Enabled

The following parameter settings apply only when **Set up PLL in LVDS Mode** is turned on for Arria GX, Stratix II, Stratix II GX, and HardCopy II fast PLLs.

When you turn on **Set up PLL in LVDS mode**, two additional options are available on the **Output Clocks** pages for c0 and c1.

The following figure shows the additional options to configure the `c0` output clock signal.

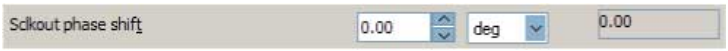
Figure 6: Additional Options to Configure the c0 Output Clock Signal



Turn on **Create sclkout0/enable0 outputs** to create the `sclkout0` and `enable0` ports. The `sclkout0` port is the serial clock output port, and the `enable0` port is the enable port.

The following figure shows the additional options to configure the `c0` output clock signal.

Figure 7: Sclkout Phase Shift Option



The `sclkout` phase shift option allows you to edit the phase shift of the `sclkout` signal (in this case, the `sclkout0` signal).

Only two pairs of `sclkout` and `enable` ports can be created in an ALTPLL IP core. The `sclkout0` and `enable0` ports are for the `c0` output clock, and the `sclkout1` and `enable1` ports are for the `c1` output clock.

Summary of PLL Output Clocks

The following table summarizes and compares properties of the clock output ports per PLL for each PLL type in the supported device families. The number of clock output ports shown in the table for each device family can be set as internal or external clock output port unless described otherwise.

Table 6: Number of Clock Output Ports per PLL

Device Family	Top_Bottom	Left_Right	Enhanced PLL	Fast PLL	Cyclone Series PLL
Arria GX	—	—	6	4	—
Arria II GX	—	7	—	—	—
Stratix IV	10	7	—	—	—
Stratix III	10	7	—	—	—
Stratix II	—	—	6	4	—

Device Family	Top_Bottom	Left_Right	Enhanced PLL	Fast PLL	Cyclone Series PLL
Stratix II GX	—	—	6	4	—
Stratix	—	—	6 (1)	3	—
Stratix GX	—	—	6 (1)	3	—
Cyclone IV	—	—	—	—	5
Cyclone III	—	—	—	—	5
Cyclone II	—	—	—	—	3
Cyclone	—	—	—	—	2 (2)

Advanced Features

Altera devices offer on-chip PLL features previously found only in high-end discrete PLL devices. These advanced features, including gated lock, clock switchover, dynamic reconfiguration, programmable bandwidth, reconfigurable bandwidth, spread spectrum clocking, and post-scale counter cascading, increase system and device performance and provide advanced clock interfacing. The following sections define each advanced feature, and describe its application, and the parameter settings you must select in the ALTPLL parameter editor to enable the feature.

Advanced Control Signals (pllena, areset, pfdena)

You can use these three signals—`pllena`, `areset`, and `pfdena`—to observe and control PLL operation and resynchronization.

pllena

Use the `pllena` signal to enable or disable the PLL. When you deassert the `pllena` signal, the PLL does not drive any output clock signal and therefore it loses lock. All counters in the PLL, including the gated lock counter, return to the default state. When you assert the `pllena` signal, the PLL drives output clock signals and tries to gain lock. The single PLL enable port on each device is shared among all PLLs on the device. By default, the `pllena` signal is tied to VCC internally.

areset

The `areset` signal is the reset or resynchronization input for each PLL. The device input pin or internal logic can drive the `areset` signal. When you assert the `areset` signal, all counters in the PLL, including the gated lock counter, are reset to initial values, in which the PLL output is cleared and the PLL is in the out-of-lock state. The VCO is also reset to its nominal setting. When the `areset` signal is deasserted, the PLL resynchronizes its input and tries to gain lock.

(1) Only four ports can be used as the external clock output ports.
(2) Only one port can be used as the external clock output port.

You should include the areset signal in your designs if any of the following conditions hold:

- PLL reconfiguration or clock switchover is enabled in your design.
- Phase relationships between the PLL input and output clocks must be maintained after a loss-of-lock condition.
- The input clock to the PLL is not toggling or is unstable at power-up.
- Assert the areset signal after the input clock is toggling while staying within the input jitter specification.

pfdena

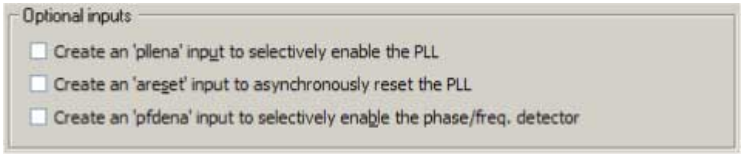
The pfdena signal enables or disables the PFD circuit. The PFD circuit is enabled by default. When the PFD circuit is disabled, the PLL output does not depend on the input clock, and tends to drift outside of the lock window. By default, the pfdena signal is tied to VCC internally.

Parameter Settings

For devices that support the advanced control signals—pllena, pfdena, and areset, the parameter settings for these signals are located on the **Inputs/Lock** or **Scan/Inputs/Lock** page of the ALTPLL parameter editor.

The following figure shows the options related to the advanced control signals. Turn on the control signal you want to create from the options available.

Figure 8: Options to Select the Advanced Control Signals



The deassertion of the pllena signal or the assertion of the areset signal does not disable the VCO, but instead resets the VCO to its nominal value. The only time that the VCO is completely disabled is when you do not have a PLL instantiated in your design.

Summary of Advanced Control Signals

The following table summarizes the device families support for advanced control signals.

Device Family	Supported Advanced Signals		
	pllena	pfdena	areset
Arria GX	Yes	Yes	Yes
Arria II GX	—	Yes	Yes
Stratix IV	—	Yes	Yes
Stratix III	—	Yes	Yes
Stratix II	Yes	Yes	Yes
Stratix II GX	Yes	Yes	Yes
Stratix	Yes	Yes	Yes

Device Family	Supported Advanced Signals		
	pllena	pfdena	areset
Stratix GX	Yes	Yes	Yes
Cyclone IV	—	Yes	Yes
Cyclone III	—	Yes	Yes
Cyclone II	Yes	Yes	Yes
Cyclone	Yes	Yes	Yes

Clock Switchover

The clock switchover feature allows the PLL to switch between two input clocks. The clock switchover feature can be used for switching between clock inputs of different frequencies and is also useful for video applications that require a manual switch between operation frequencies. The clock switchover capability is widely implemented in telecommunication, storage, and server markets because these markets require highly reliable clocking schemes to ensure system reliability.

The following clock switchover modes are supported by the ALTPLL IP core:

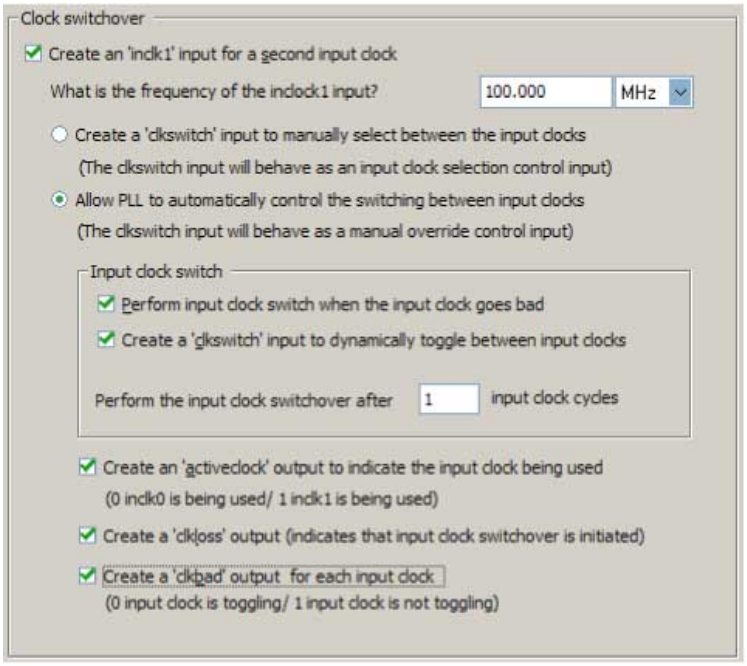
- **Automatic switchover**—The PLL monitors the currently used clock signal, and if it stops toggling or loss-of-lock occurs, the PLL automatically switches to the other clock signal (`inclk0` or `inclk1`).
- **Manual clock switchover**—The clock switchover is controlled using the `clkswitch` signal. The manual override feature available in automatic clock switchover is different from the manual clock switchover.

Parameter Settings

For devices that support the clock switchover feature, the parameter settings are located on the **Clock switchover** page of the ALTPLL parameter editor.

The following figure shows all the options available on the **Clock switchover** page for an Arria GX device. The options are device-dependent and what you see may differ.

Figure 9: Clock Switchover



To enable the switchover feature, turn on **Create an 'inclk1' input for a second input clock**, and specify the frequency of the `inclk1` signal. The `inclk0` signal is by default the primary input clock signal of the ALTPLL IP core.

Select the related option for manual or automatic clock switchover mode. For the automatic clock switchover mode, you can choose to create the `clkswitch` signal as a manual override. The automatic switchover is initiated during loss of lock or when the `inclk0` signal stops toggling or when the `clkswitch` signal is asserted. You must specify the number of clock cycles to wait before the PLL performs the clock switchover. Note that the allowed number of clock cycles to wait is device-dependant.

You can use the optional signals – `activeclock`, `clkloss`, and `clkbad`—as indicators when you use the clock switchover feature.

Use the `activeclock` signal to monitor which input clock signal is driving the PLL. When the current clock signal is `inclk0`, the `activeclock` signal is low. When the current clock signal is `inclk1`, the `activeclock` signal is high.

Use the `clkbad` signals (`clkbad0` and `clkbad1`) to monitor which input clock signal has stopped toggling. The `clkbad0` signal is used to monitor the `inclk0` signal, and the `clkbad1` signal monitors the `inclk1` signal. The `clkbad0` signal goes high when the `inclk0` signal stops toggling, and the `clkbad1` signal goes high when the `inclk1` signal stops toggling. The `clkbad` signals remain low when the input clock signals are toggling.

Use the `clkloss` signal to monitor the current status of the clock switchover. The `clkloss` signal goes high to indicate that loss of lock has been detected, and the clock switchover is initiated. The `clkloss` signal remains low when the clock switchover is not initiated. The `clkloss` signal is only available in Arria GX, Stratix, Stratix GX, Stratix II, and Stratix II GX devices.

The following top-level ports are created from these parameter settings:

- Input ports: `inclk1` and `clkswitch`.
- Output ports: `activeclock`, `clkloss`, `clkbad0`, and `clkbad1`.

Summary of Automatic Clock Switchover Feature

The following table summarizes automatic clock switchover support in the supported device families. Also supports automatic clock switchover feature with manual override control.

Table 7: Automatic Clock Switchover Feature Support

Device Family	Top_Bottom	Left_Right	Enhanced PLL	Fast PLL	Cyclone Series PLL
Arria GX	—	—	Yes	No	—
Arria II GX	—	Yes	—	—	—
Stratix IV	Yes	Yes	—	—	—
Stratix III	Yes	Yes	—	—	—
Stratix II	—	—	Yes	No	—
Stratix II GX	—	—	Yes	No	—
Stratix	—	—	Yes	No	—
Stratix GX	—	—	Yes	No	—
Cyclone IV	—	—	—	—	Yes
Cyclone III	—	—	—	—	Yes
Cyclone II	—	—	—	—	No
Cyclone	—	—	—	—	No

The following table summarizes the manual clock switchover support in the supported device families.

Table 8: Manual Clock Switchover Feature Support

Device	Top_Bottom	Left_Right	Enhanced PLL	Fast PLL	Cyclone Series PLL
Arria GX	—	—	Yes	Yes	—
Arria II GX	—	Yes	—	—	—
Stratix IV	Yes	Yes	—	—	—
Stratix III	Yes	Yes	—	—	—
Stratix II	—	—	Yes	Yes	—
Stratix II GX	—	—	Yes	Yes	—
Stratix	—	—	Yes	No	—
Stratix GX	—	—	Yes	No	—
Cyclone IV	—	—	—	—	Yes

Device	Top_Bottom	Left_Right	Enhanced PLL	Fast PLL	Cyclone Series PLL
Cyclone III	—	—	—	—	Yes
Cyclone II	—	—	—	—	Yes
Cyclone	—	—	—	—	No

Spread-Spectrum Clocking

Spread-spectrum technology reduces electromagnetic interference (EMI) in a system. This technology works by distributing the clock energy over a broad frequency range.

The spread-spectrum clocking feature distributes the fundamental clock frequency energy throughout your design to minimize energy peaks at specific frequencies. By reducing the spectrum peak amplitudes, the feature makes your design more likely meets the EMI emission compliance standards, and reduces costs associated with traditional EMI containment.

The traditional methods for limiting EMI include shielding, filtering, and using multi-layer printed circuit boards. Multi-layer circuit boards are expensive and are not guaranteed to meet the EMI emission compliance standards. The use of spread-spectrum technology is simpler and more cost-effective than these other methods.

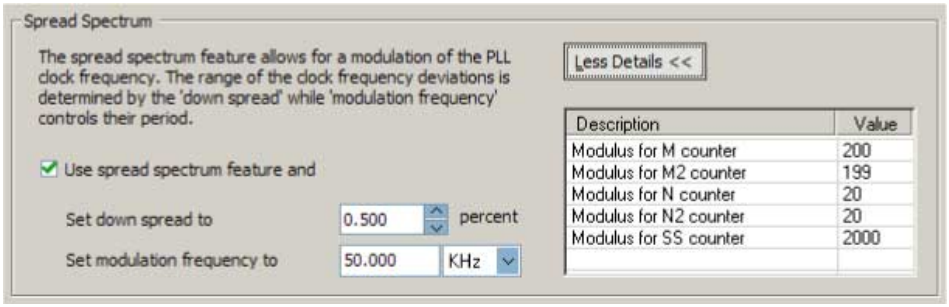
To use the spread-spectrum clocking feature, you must set the programmable bandwidth feature to **Auto**.

Parameter Settings

For devices that support spread-spectrum technology, the parameter settings are located on the **Bandwidth/SS** page of the ALTPLL parameter editor.

The following figure shows the Spread Spectrum window.

Figure 10: Spread Spectrum Settings



To enable the spread-spectrum feature, turn on **Use spread spectrum feature**. Set the desired down spread percentage, and the modulation frequency. The table in the spread spectrum window lists the detailed descriptions of the current counter values.

The down spread percentage defines the modulation width or frequency span of the instantaneous output frequency resulting from the spread spectrum. When you use down spread, the modulation width falls at or below a specified maximum output frequency. The wider the modulation, the larger the band of frequencies over which the energy is distributed, and the more reduction is achieved from the peak. For example, with a down spread percentage of 0.5% and maximum operating frequency of 100 MHz, the output frequency is swept between 99.5 and 100 MHz.

The modulation frequency, often called sweep rate, defines how fast the spreading signal sweeps from the minimum to the maximum frequency.

Turning on spread-spectrum clocking creates no new top-level ports.

Summary of Spread-Spectrum Clocking Feature

The following table summarizes the PLL types that support the spread-spectrum clocking feature.

Table 9: Spread-Spectrum Clocking Feature

Device	Top_Bottom	Left_Right	Enhanced PLL	Fast PLL	Cyclone Series PLL
Stratix IV	Yes (3)	Yes (3)	—	—	—
Cyclone IV	—	—	—	—	No

Gated Lock and Self-Reset

The lock time of a PLL is defined as the amount of time required by the PLL to attain the target frequency and phase relationship after device power-up, after a change in the PLL output frequency, or after resetting the PLL.

A PLL might lose lock for a number of reasons, such as the following causes:

- Excessive jitter on the input clock.
- Excessive switching noise on the clock inputs of the PLL.
- Excessive noise from the power supply can cause high output jitter and possible loss of lock.
- A glitch or stopping of the input clock to the PLL.
- Resetting the PLL by asserting the `areset` or `pllena` ports of the PLL.
- An attempt to reconfigure the PLL might cause the M counter, N counter, or phase shift to change, which causes the PLL to lose lock. However, changes to the post-scale counters do not affect the PLL locked signal.
- PLL input clock frequency drifts outside the lock range specification.
- The PFD is disabled using the `pfdena` port. When this happens, the PLL output phase and frequency tend to drift outside of the lock window.

The ALTPLL IP core allows you to monitor the PLL locking process using a lock signal named `locked` and also allows you to set the PLL to self-reset on loss of lock.

Gated Lock

Some devices support a gated lock signal that allows you to configure a programmable 20-bit counter that holds the lock signal low for a user-specified number of input clock transitions. This is useful to eliminate the false toggling of the lock signal as the PLL begins tracking the reference clock. Gated lock allows the PLL to lock before asserting the `locked` signal, providing a stabilized lock signal.

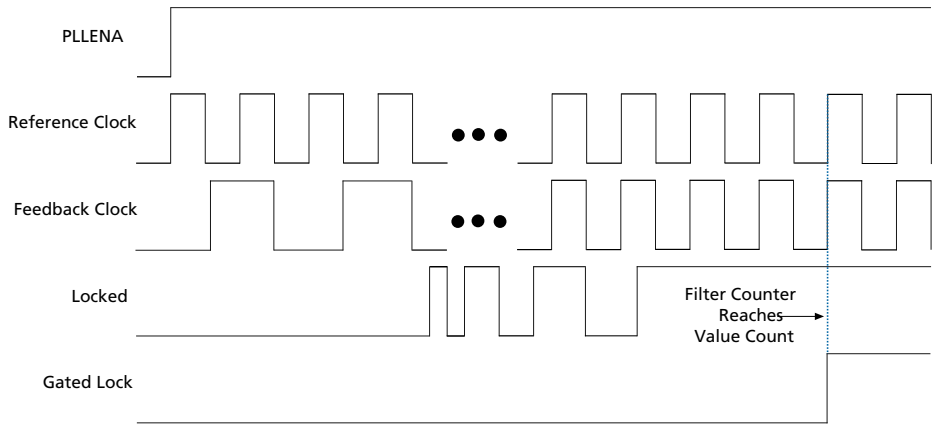
An asserted `locked` signal indicates the PLL clock output is aligned with the PLL reference input clock. The `locked` signal might toggle as the PLL begins tracking the reference clock. To avoid such a false lock indication,

(3) This device can accept a spread-spectrum input with typical modulation frequencies, but it cannot generate spread-spectrum clock signals internally.

use a gated lock signal. A gated `locked` signal or an ungated `locked` signal can feed a logic array or an output pin. When you must reset the gated counter, reset the PLL by asserting the `areset` signal or the `pllena` signal.

The following figure shows the timing waveform for gated and ungated locked signals.

Figure 11: Input and Output Ports



Self-Reset on Loss of Lock

This feature allows the PLL to self-reset upon loss of lock, normally for the same reasons described in **Gated Lock and Self-Reset** on page 18

Related Information

Gated Lock and Self-Reset on page 18

Parameter Settings

To enable the `locked` signal, and the self-reset feature in the ALTPLL IP core, use the parameter settings on the **Scan/Inputs/Lock** or **Inputs/Lock** page of the ALTPLL parameter editor.

The following figure shows the related options in the **Scan/Inputs/Lock** or **Inputs/Lock** page for an Arria II GX device. Note that the options are device-dependent and what you see may differ.

Figure 12: Lock Output Options

Turning on **Create 'locked' output** creates an output port named `locked` in the ALTPLL IP core.

Turning on **Enable self-reset on loss of lock** enables the self-reset feature.

In devices that support gated lock, another option appears on the page, which is the **Hold 'locked' output** option. Turning on this option enables the gated lock circuitry to gate the `locked` signal. You must specify the number of PLL input clock cycles to hold the `locked` signal low after the PLL is initialized. This value is used by the gated lock counter. The value ranges from **1** to **1,048,575** clock cycles.

The following figure shows the **Hold Locked Output** option.

Figure 13: Hold Locked Output Option

Calculating the Value of Gated Lock Counter

To calculate the number of clock cycles needed, you must know the maximum lock time of the PLL, and the period of the PLL input clock. The lock time of the PLL is listed in the "PLL Timing Specifications" section of the *DC & Switching Characteristics* chapter of the device handbook. The period of the PLL input clock is user-specified. For example, if the maximum lock time of a PLL is 1 ms, and its input clock frequency is 100 MHz which corresponds to a 10 ns clock period, you calculate the value of the gated lock counter, by dividing 1 ms by 10 ns. The result is 100,000 clock cycles.

Only the `locked` port is created from these parameter settings.

Summary of Gated Lock Signals and Self-Reset on Loss of Lock

The following table summarizes the device families that support the gated lock and self-reset on loss of lock features.

Table 10: Gated Lock Signals and Self-Reset on Loss of Lock

Device Family	Gated Lock Support	Self-Reset on Loss of Lock
Arria GX	Yes	—
Arria II GX	—	Yes
Stratix IV	—	Yes
Stratix III	—	Yes
Stratix II	Yes	—
Stratix II GX	Yes	—
Stratix	—	—
Stratix GX	—	—
Cyclone IV	—	Yes
Cyclone III	—	Yes
Cyclone II	Yes	—
Cyclone	—	—

Programmable Bandwidth

The PLL bandwidth is defined as the ability of the PLL to track the input clock and jitter. The bandwidth is measured by the -3 dB frequency of the closed-loop gain in the PLL, or approximately the unity gain point of the PLL open loop response. Altera devices provide a programmable PLL bandwidth feature that allows

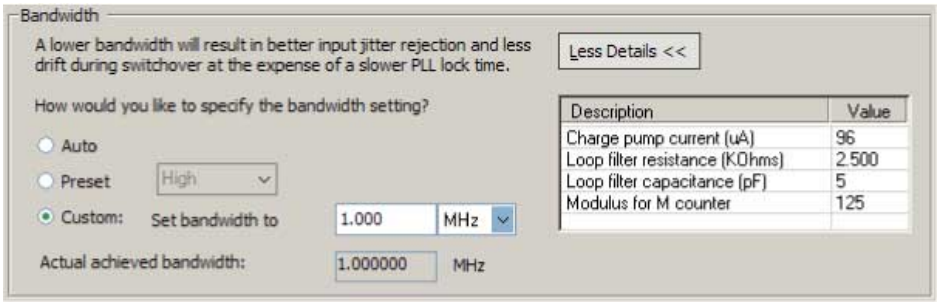
you to configure the characteristics of the PLL loop filter. Most loop filters contain only passive components, such as resistors and capacitors, which consumes board space. Altera FPGAs already contain these components, and by using the programmable bandwidth feature, you can control how the components affect the PLL bandwidth. This includes controlling the charge pump current, loop filter resistance, and high frequency capacitance values. The charge pump current affects the PLL bandwidth directly. The higher the charge pump current, the higher the PLL bandwidth.

Parameter Settings

The parameter settings to configure the bandwidth of the ALTPLL IP core are located on the **Bandwidth/SS** page of the ALTPLL parameter editor.

The following figure shows the bandwidth configuration options on the **Bandwidth/SS** page.

Figure 14: Bandwidth Configuration Options



The following list describes the preset values that you can choose:

- **Low**—PLL with a low bandwidth has better jitter rejection but a slower lock time.
- **High**—PLL with a high bandwidth has a faster lock time but tracks more jitter.
- **Medium**—A medium bandwidth offers a balance between lock time and jitter rejection.

If you select **Auto**, the ALTPLL parameter editor chooses the best possible bandwidth values to achieve the desired PLL settings. In some cases, you can get a bandwidth value outside the **Low** and **High** preset range.

To set the bandwidth manually, select **Custom**, and specify the value. The compiler attempts to achieve the value that you specify, or the closest possible value to achieve your desired setting. You can check the bandwidth value in the compilation report.

The table on the right in the **Bandwidth/SS** page shows the values of the charge pump current, loop filter resistance and capacitance, and the *M* counter.

An advanced level of control is also possible for precise control of the PLL loop filter characteristics. This level allows you to select the charge pump current, and loop filter resistance and capacitance values explicitly. The advanced parameters are: `charge_pump_current`, `loop_filter_r`, and `loop_filter_c`.

You can use the programmable bandwidth feature with the clock switchover or spread-spectrum features to get the PLL output settings that you desire. You must set the bandwidth to **Auto** if you want to enable the spread-spectrum feature.

These parameter settings create no additional top-level ports.

Summary of Programmable Bandwidth Support

The following table summarizes programmable bandwidth support in the different device families.

Table 11: Programmable Bandwidth Support

Device Family	Top_Bottom	Left_Right	Enhanced PLL	Fast PLL	Cyclone Series PLL
Arria GX	—	—	Yes	Yes	—
Arria II GX	—	Yes (4)	—	—	—
Stratix IV	Yes (4)	Yes (4)	—	—	—
Stratix III	Yes (4)	Yes (4)	—	—	—
Stratix II	—	—	Yes	Yes	—
Stratix II GX	—	—	Yes	Yes	—
Stratix	—	—	Yes	No	—
Stratix GX	—	—	Yes	No	—
Cyclone IV	—	—	—	—	Yes (4)
Cyclone III	—	—	—	—	Yes (4)
Cyclone II	—	—	—	—	No
Cyclone	—	—	—	—	No

Advanced PLL Parameters

The ALTPLL parameter editor provides an option for you to generate output files using the ALTPLL advanced parameters. Examples of advanced parameters are `charge_pump_current`, `loop_filter_r`, and `loop_filter_c`. This option is supported in all Altera devices.

This option is intended for advanced users who know the exact details of their PLL configuration and for users who understand the parameters well enough to set them optimally. The files generated are not intended to be re-used by the ALTPLL parameter editor, because after the ALTPLL IP core output files are specified using the advanced parameters, the Quartus II compiler cannot change them. For example, the compiler cannot perform optimization. Thus, your design cannot benefit from improved algorithms to pick better settings or to make changes to some settings that the ALTPLL parameter editor finds to be incompatible with your design.

Advanced Parameter Settings

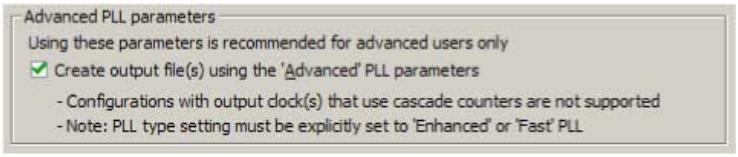
Altera recommends you not to use these parameters unless you are an advanced user.

(4) This device family supports only **Auto** and preset values.

The parameter settings to generate output files using advanced PLL parameters are located on the **Scan/Inputs/Lock or Inputs/Lock** page of the ALTPLL parameter editor.

The following figure shows the Advanced PLL parameters window.

Figure 15: Advanced PLL Parameters



Turn on **Create output file(s) using the 'Advanced' PLL parameters** to enable the feature. You must explicitly select enhanced or fast PLL to use this option.

When this option is turned on, the generated output files contain all of the initial counter values used in the PLL. You can use these values for functional simulation in a third-party simulator.

These parameter settings create no additional top-level ports.

PLL Dynamic Reconfiguration

The PLL Dynamic Reconfiguration feature allows you to reconfigure your PLL on-the-fly. You can control the configuration process using the following ports:

- Input ports: `scanclk`, `scandata`, `scanclkena`, and `configupdate`.
- Output ports: `scandataout` and `scandone`. The `scandone` port is not available for Cyclone, Cyclone II, Stratix, and Stratix GX devices.

The Stratix and Stratix GX enhanced PLLs can be dynamically reconfigured using scan chains. Depending on the PLL type, two options are available for the scan chain – long or short. The long scan chain allows the configuration of those PLLs with six core and four external clocks, while the short scan chain limits the configuration to those PLLs with the six core clocks with no external clocks.

The scan chain method for dynamic reconfiguration is not available for all supported device families. The devices that support the normal dynamic reconfiguration scheme uses configuration files, such as the Hexadecimal-format file, **.hex**, or the Memory Initialization file, **.mif**. These files are used together with the ALTPLL_RECONFIG IP core to perform the dynamic configuration.

Related Information

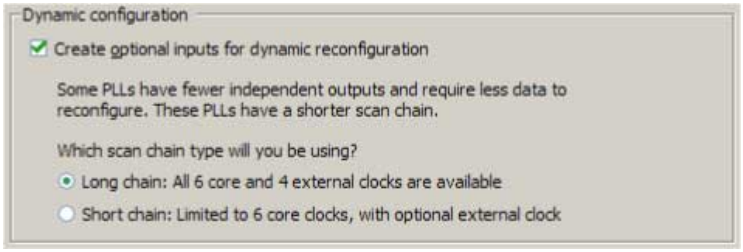
Phased-Locked Loop Reconfiguration (ALTPLL_RECONFIG) User Guide.

Parameter Settings

To enable the dynamic reconfiguration feature for Stratix and Stratix GX devices, use the parameter settings located on the **Scan/Input/Lock** page of the ALTPLL parameter editor.

The following figure shows the dynamic reconfiguration options using scan chains.

Figure 16: Dynamic Reconfiguration Options for Stratix and Stratix GX Devices



The following table lists the dynamic reconfiguration options using scan chains.

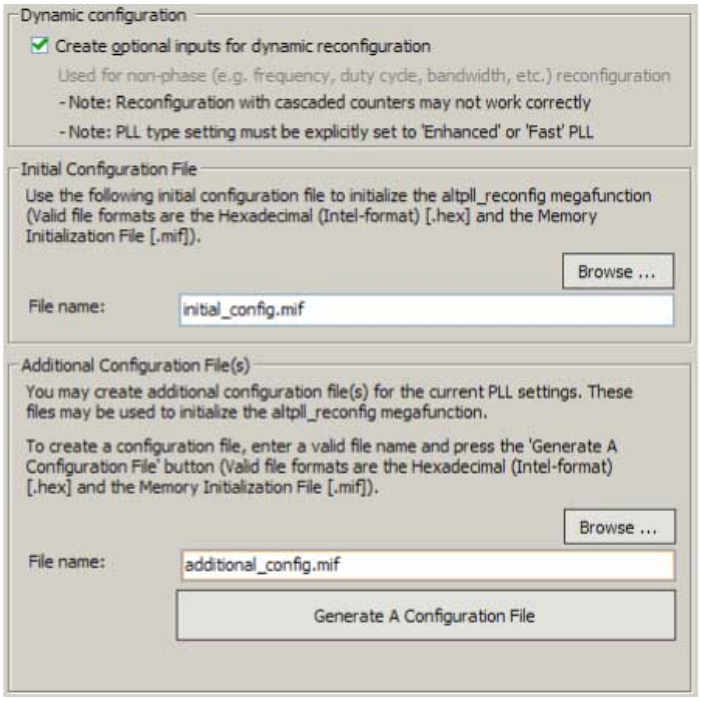
Table 12: Dynamic Reconfiguration Options for Stratix and Stratix GX Devices

Option	Description
Create optional inputs for dynamic reconfiguration	This option enables all the PLL reconfiguration ports for this instantiation—`scanclk`, `scanaclr`, `scandata`, and `scandataout`.
Which scan chain type will you be using?	This option lets you specify the following types of scan chain to be used in the PLL reconfiguration: • **Long chain**—Specifies that you are using PLLs 5 and 6 with PLL reconfiguration. PLLs 5 and 6 have six logic array outputs and four external clock outputs and therefore, have a longer reconfiguration chain. • **Short chain**—Specifies that you are using PLLs 11 and 12 with PLL reconfiguration. PLLs 11 and 12 have only six logic array outputs with no dedicated external clock output counters and are considered the shorter reconfiguration chain PLLs.

For devices that use the normal dynamic reconfiguration scheme, the parameter settings are located on the **PLL Reconfiguration** page of the ALTPLL parameter editor.

The following figure shows the normal dynamic configuration scheme options that uses configuration files.

Figure 17: Dynamic Reconfiguration Options Using Configuration Files



The following table lists the normal dynamic reconfiguration scheme options.

Table 13: Normal Dynamic Reconfiguration Options and Descriptions

Option	Description
Create optional inputs for dynamic reconfiguration	This option enables all the PLL reconfiguration ports for this instantiation—`scanclk`, `scanaclr`, `scandata`, `scandone`, and `scandataout`.
Initial Configuration File	Specify the location of the configuration file that is used to initialize the ALTPLL_RECONFIG IP core.
Additional Configuration File(s)	Specify additional configuration file. This file might contain additional settings for the PLL, or might be used to initialize the ALTPLL_RECONFIG IP core.

Summary of Supported Device Families for PLL Dynamic Reconfiguration Feature

The following table summarizes PLL dynamic reconfiguration support in the different device families and PLL types.

Table 14: PLL Dynamic Reconfiguration Feature Support

Device Family	Top_Bottom	Left_Right	Enhanced PLL	Fast PLL	Cyclone Series PLL
Arria GX	—	—	Yes	Yes	—
Arria II GX	—	Yes	—	—	—

Device Family	Top_Bottom	Left_Right	Enhanced PLL	Fast PLL	Cyclone Series PLL
Stratix IV	Yes	Yes	—	—	—
Stratix III	Yes	Yes	—	—	—
Stratix II	—	—	Yes	Yes	—
Stratix II GX	—	—	Yes	Yes	—
Stratix	—	—	Yes (5)	No	—
Stratix GX	—	—	Yes (5)	No	—
Cyclone IV	—	—	—	—	Yes
Cyclone III	—	—	—	—	Yes
Cyclone II	—	—	—	—	No
Cyclone	—	—	—	—	No

Dynamic Phase Configuration

The dynamic phase configuration feature allows the output phases of individual PLL outputs to dynamically adjust relative to each other and the reference clock, without sending serial data through the scan chain of the corresponding PLL. This feature is also known as the dynamic phase stepping feature.

You can use this feature to quickly adjust the output clock phase shift in real time. This adjustment is achieved by incrementing or decrementing the VCO phase-tap selection to a C counter or to the M counter. By default, the phase is shifted by 1/8th of the VCO frequency at each step. However, you can easily modify the phase shift step resolution of the individual PLL output clock using the ALTPLL parameter editor.

For dynamic phase shifting to work correctly, the PLL must have the following ports:

- Input ports: `phasecounterselect[3..0]`, `phaseupdown`, `phasestep`, and `scanclk`
- Output port: `phasedone`

Related Information

Ports and Parameters on page 38

Implementing PLL Reconfiguration in Stratix III and Stratix IV Devices
This application note describes the flow for implementing phase-locked loop (PLL) reconfiguration in Stratix III and Stratix IV devices.

Supported Devices

The dynamic phase configuration feature is available only for Cyclone IV, Cyclone III, Arria II GX, Stratix III, and Stratix IV device families.

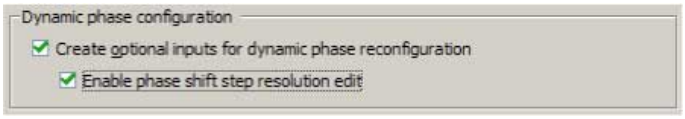
(5) This device family supports dynamic reconfiguration using scan chains only.

Parameter Settings

The parameter settings to enable the dynamic phase configuration feature are located on the **PLL Reconfiguration** page of the ALTPLL parameter editor.

The following figure shows the dynamic phase configuration options.

Figure 18: Dynamic Phase Configuration



Turning on **Create optional inputs for dynamic phase reconfiguration** enables the feature, and create these ports - `phasecounterselect[3..0]`, `phaseupdown`, `phasestep`, `scanclk`, and `phasedone`.

Turning on **Enable phase shift step resolution edit** allows you to modify the phase shift step resolution value for each individual PLL output clock on its **Output Clocks** page.

The following figure shows the new option that appears on each output clock settings page. By default, the finest phase shift resolution value is 1/8th of the VCO period. If the VCO frequency is at the lower end of the supported VCO range, the phase shift resolution might be larger than you would prefer for your design. Use this option to fine tune the phase shift step resolution.

Figure 19: Phase Shift Step Configuration

Modifying the PLL Phase Shift Step Resolution Using Advanced Parameters

The finest phase shift step resolution you can get in the ALTPLL IP core is 1/8th of the VCO period. If the VCO frequency is at the lower end of the supported VCO range, the phase shift step resolution may be larger than you would prefer for your design.

You can modify your phase shift resolution using the dynamic phase reconfiguration feature of the PLL. If you want to modify the phase shift resolution without the dynamic phase reconfiguration feature enabled, perform the following steps:

1. Create an ALTPLL instance. Make sure you specify the speed grade of your target device and the PLL type.
2. On the **PLL Reconfiguration** page, turn on **Create Optional Inputs for Dynamic Phase Reconfiguration** and **Enable Phase Shift Step Resolution Edit**.
3. On the **Output Clocks** page, set your desired phase shift for each required output clock. Click **More Details** to see the internal PLL settings. Note all of the settings shown.
4. On the **Bandwidth/SS** page, click **More Details** to see the internal PLL settings. Note all of the settings shown.
5. On the **Inputs/Lock** page, turn on **Create output file(s) using the 'Advanced' PLL Parameters**.
6. Return to the **PLL Reconfiguration** page and turn off **Create Optional Inputs for Dynamic Phase Reconfiguration**.
7. Click **Finish** to generate the PLL instantiation file(s).

When using Advanced Parameters, the PLL wrapper file (<*ALTPLL_instantiation_name*>**.v|.vhd**) is written in a format that allows you to identify the PLL parameters. The parameters are listed in the **Generic Map** section of the VHDL file, or in the `defparam` section of the Verilog file.

8. Open your PLL instantiation wrapper file and locate either the **Generic Map** or the **defparam** section.
9. Modify the settings to match the settings that you noted in steps 3 and 4.
10. Save the PLL instantiation wrapper file and compile your design.
11. Verify that the output clock frequencies and phases are correct in the PLL Usage report located under Resource Section of the Fitter folder in the Compilation Report.

By using this technique, you can apply valid PLL parameters as provided by the ALTPLL parameter editor to optimize the settings for your design.

Alternatively, you can leave the dynamic phase reconfiguration option enabled and tie the relevant input ports—`phasecounterselect[3..0]`, `phaseupdown`, `phasestep`, and `scanclk`—to constants, if you prefer not to manually edit the PLL wrapper file using Advanced PLL Parameters option.

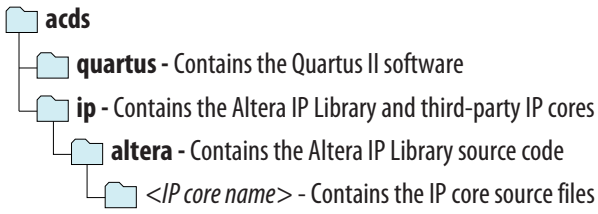
PLL Output Counter Cascading

In 28 nm devices, a C-counter input can be either a VCO output or the cascaded output of a neighboring C-counter. Cascading C-counters increase the possible division factor, enabling very low frequency PLL output clocks.

Installing and Licensing IP Cores

The Altera IP Library provides many useful IP core functions for production use without purchasing an additional license. You can evaluate any Altera IP core in simulation and compilation in the Quartus II software using the OpenCore evaluation feature. Some Altera IP cores, such as MegaCore® functions, require that you purchase a separate license for production use. You can use the OpenCore Plus feature to evaluate IP that requires purchase of an additional license until you are satisfied with the functionality and performance. After you purchase a license, visit the Self Service Licensing Center to obtain a license number for any Altera product.

Figure 20: IP Core Installation Path



Note: The default IP installation directory on Windows is **<*drive*>:\altera**<*version number*>; on Linux it is <*home directory*>**/altera/** <*version number*>.

Related Information

- **Altera Licensing Site**
- **Altera Software Installation and Licensing Manual**

Customizing and Generating IP Cores

You can customize IP cores to support a wide variety of applications. The Quartus II IP Catalog and parameter editor allow you to quickly select and configure IP core ports, features, and output files.

IP Catalog and Parameter Editor (replaces MegaWizard Plug-In Manager)

The Quartus II IP Catalog (**Tools** > **IP Catalog**) and parameter editor help you easily customize and integrate IP cores into your project. You can use the IP Catalog and parameter editor to select, customize, and generate files representing your custom IP variation.

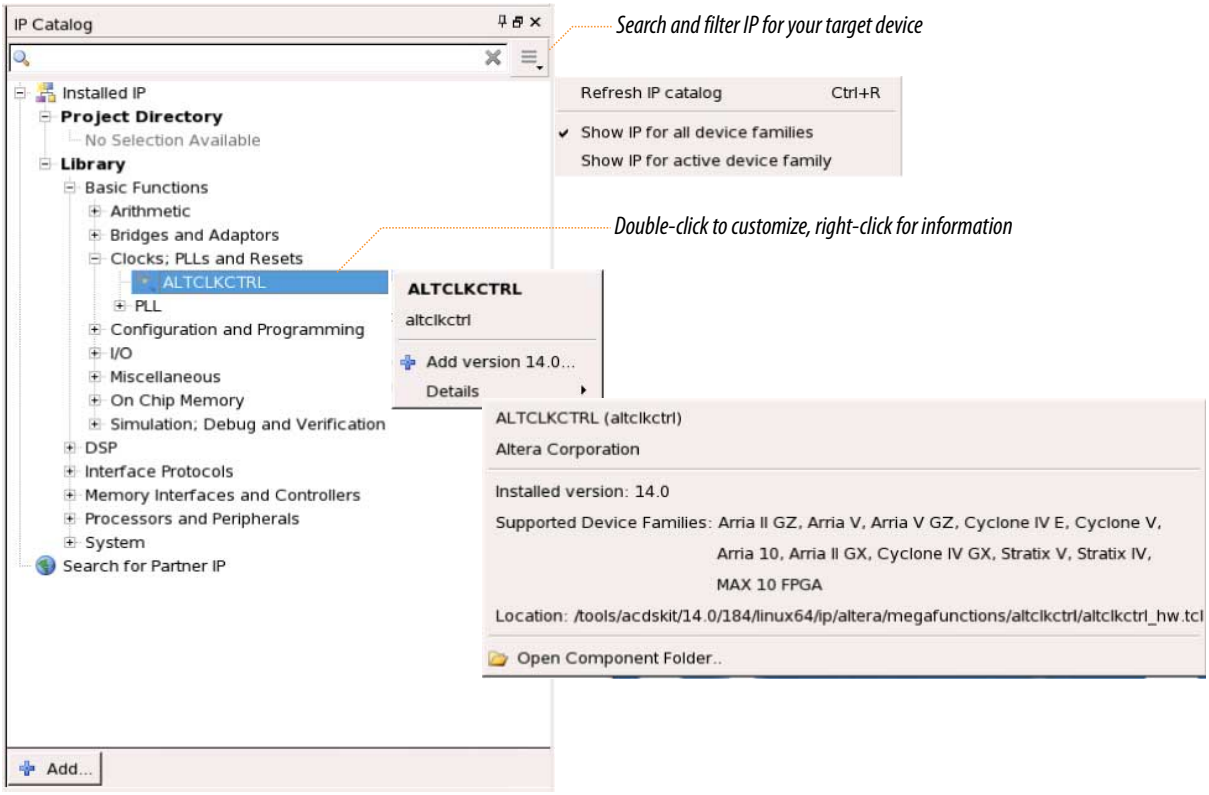
Note: The IP Catalog (**Tools** > **IP Catalog**) and parameter editor replace the MegaWizard™ Plug-In Manager for IP selection and parameterization, beginning in Quartus II software version 14.0. Use the IP Catalog and parameter editor to locate and paramaterize Altera IP cores.

The IP Catalog lists IP cores available for your design. Double-click any IP core to launch the parameter editor and generate files representing your IP variation. The parameter editor prompts you to specify an IP variation name, optional ports, and output file generation options. The parameter editor generates a top-level Qsys system file (**.qsys**) or Quartus II IP file (**.qip**) representing the IP core in your project. You can also parameterize an IP variation without an open project.

Use the following features to help you quickly locate and select an IP core:

- Filter IP Catalog to **Show IP for active device family** or **Show IP for all device families**.
- Search to locate any full or partial IP core name in IP Catalog. Click **Search for Partner IP**, to access partner IP information on the Altera website.
- Right-click an IP core name in IP Catalog to display details about supported devices, open the IP core's installation folder, andor view links to documentation.

Figure 21: Quartus II IP Catalog



Note: The IP Catalog is also available in Qsys (**View** > **IP Catalog**). The Qsys IP Catalog includes exclusive system interconnect, video and image processing, and other system-level IP that are not available in the Quartus II IP Catalog. For more information about using the Qsys IP Catalog, refer to *Creating a System with Qsys* in the *Quartus II Handbook*.

Related Information

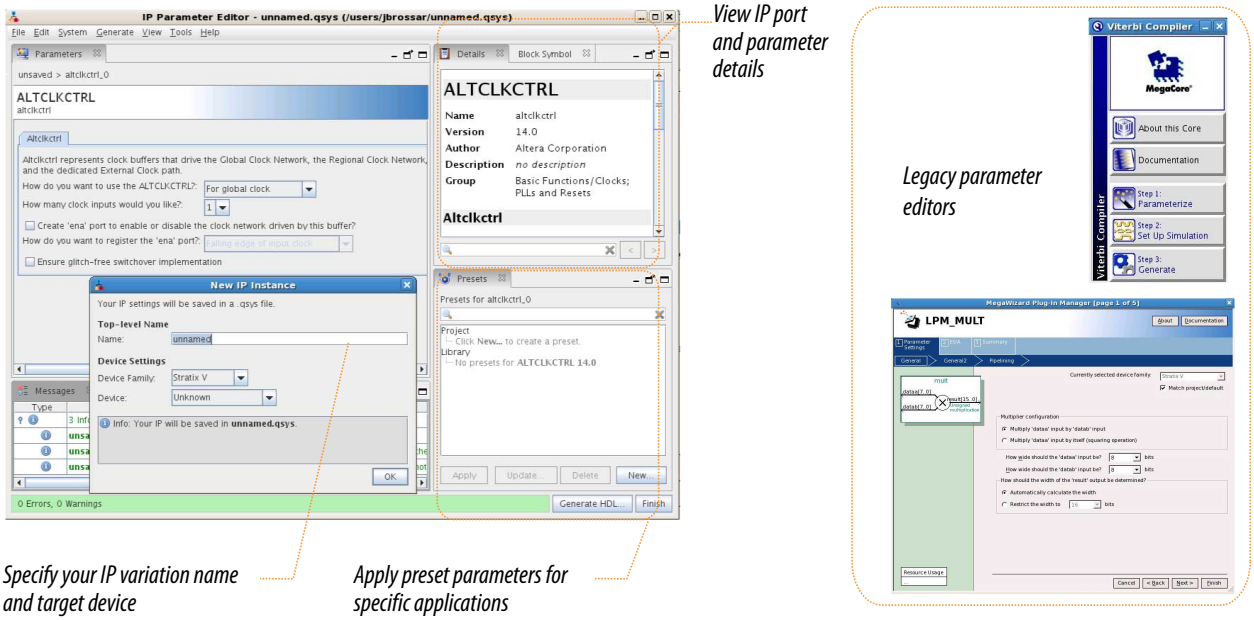
- **Creating a System with Qsys**

Using the Parameter Editor

The parameter editor helps you to configure IP core ports, parameters, and output file generation options.

- Use preset settings in the parameter editor (where provided) to instantly apply preset parameter values for specific applications.
- View port and parameter descriptions, and links to documentation.
- Generate testbench systems or example designs (where provided).

Figure 22: IP Parameter Editors

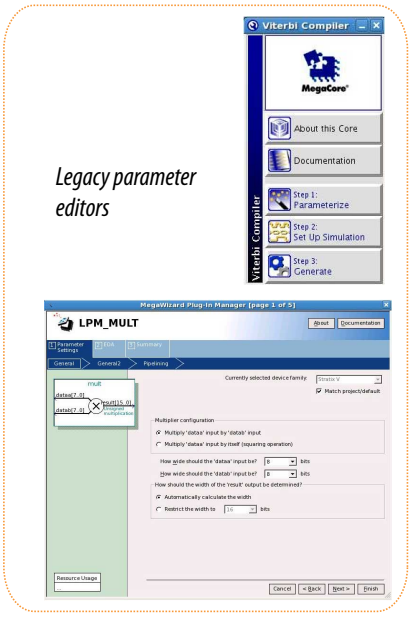


Specifying IP Core Parameters and Options (Legacy Parameter Editors)

The Quartus II software version 14.0 and previous uses a legacy version of the parameter editor for IP core configuration and generation. Use the following steps to configure and generate an IP variation using a legacy parameter editor.

Note: The legacy parameter editor generates a different output file structure than the latest parameter editor. Refer to *Specifying IP Core Parameters and Options* for configuration of IP cores in the Quartus II software version 14.0a10 and later.

Figure 23: Legacy Parameter Editors



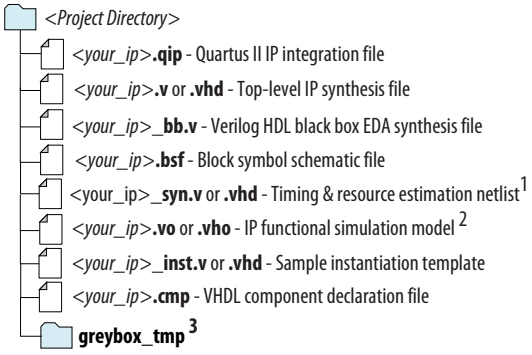
1. In the IP Catalog (**Tools** > **IP Catalog**), locate and double-click the name of the IP core to customize. The parameter editor appears.
2. Specify a top-level name and output HDL file type for your IP variation. This name identifies the IP core variation files in your project. Click **OK**.
3. Specify the parameters and options for your IP variation in the parameter editor. Refer to your IP core user guide for information about specific IP core parameters.
4. Click **Finish** or **Generate** (depending on the parameter editor version). The parameter editor generates the files for your IP variation according to your specifications. Click **Exit** if prompted when generation is complete. The parameter editor adds the top-level **.qip** file to the current project automatically.

   Note: To manually add an IP variation generated with legacy parameter editor to a project, click **Project** > **Add/Remove Files in Project** and add the IP variation **.qip** file.

Files Generated for Altera IP Cores (version 14.0 and previous)

The Quartus II version 14.0 and earlier software generates the following output for your IP core.

Figure 24: IP Core Generated Files



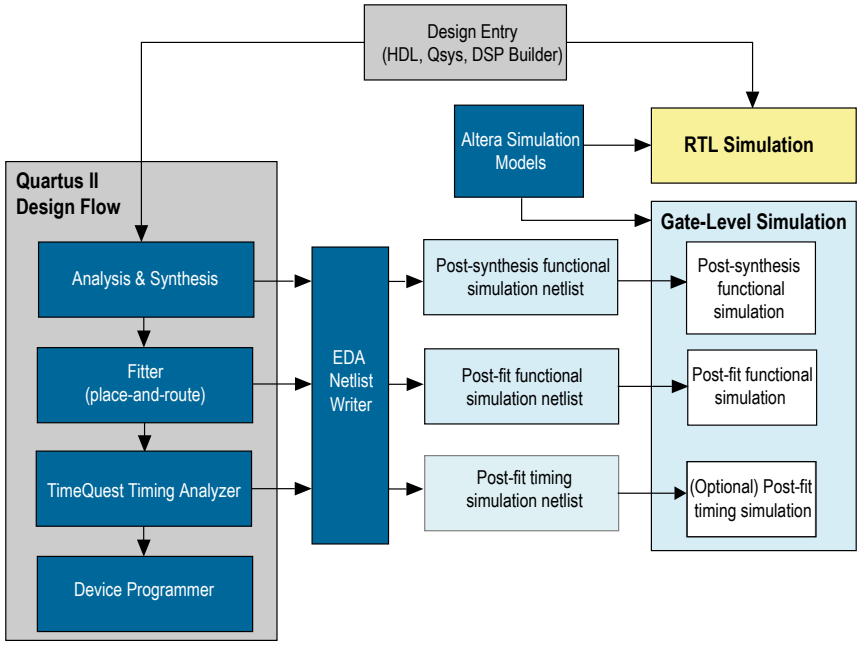
Notes:
1. If supported and enabled for your IP variation
2. If functional simulation models are generated
3. Ignore this directory

Simulating Altera IP Cores in other EDA Tools

The Quartus II software supports RTL and gate-level design simulation of Altera IP cores in supported EDA simulators. Simulation involves setting up your simulator working environment, compiling simulation model libraries, and running your simulation.

You can use the functional simulation model and the testbench or example design generated with your IP core for simulation. The functional simulation model and testbench files are generated in a project subdirectory. This directory may also include scripts to compile and run the testbench. For a complete list of models or libraries required to simulate your IP core, refer to the scripts generated with the testbench. You can use the Quartus II NativeLink feature to automatically generate simulation files and scripts. NativeLink launches your preferred simulator from within the Quartus II software.

Figure 25: Simulation in Quartus II Design Flow



Note: Post-fit timing simulation is not supported for 28nm and later device archetectures. Altera IP supports a variety of simulation models, including simulation-specific IP functional simulation models and encrypted RTL models, and plain text RTL models. These are all cycle-accurate models. The models support fast functional simulation of your IP core instance using industry-standard VHDL or Verilog HDL simulators. For some cores, only the plain text RTL model is generated, and you can simulate that model. Use the simulation models only for simulation and not for synthesis or any other purposes. Using these models for synthesis creates a nonfunctional design.

Related Information

Simulating Altera Designs

Upgrading IP Cores

IP core variants generated with a previous version of the Quartus II software may require upgrading before use in the current version of the Quartus II software. Click **Project** > **Upgrade IP Components** to identify and upgrade IP core variants.

The **Upgrade IP Components** dialog box provides instructions when IP upgrade is required, optional, or unsupported for specific IP cores in your design. You must upgrade IP cores that require it before you can compile the IP variation in the current version of the Quartus II software. Many Altera IP cores support automatic upgrade.

The upgrade process renames and preserves the existing variation file (**.v**, **.sv**, or **.vhd**) as *<my_variant>*_**BAK.v**, **.sv**, **.vhd** in the project directory.

Table 15: IP Core Upgrade Status

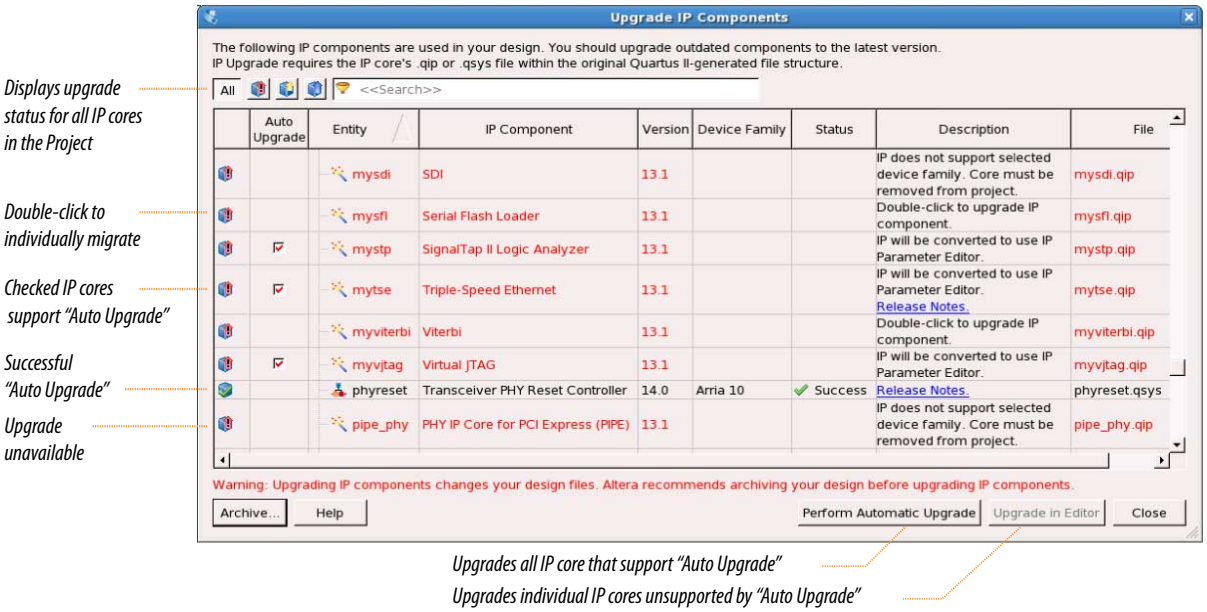
IP Core Status	Corrective Action
Required Upgrade IP Components	You must upgrade the IP variation before compiling in the current version of the Quartus II software.
Optional Upgrade IP Components	Upgrade is optional for this IP variation in the current version of the Quartus II software. You can upgrade this IP variation to take advantage of the latest development of this IP core. Alternatively you can retain previous IP core characteristics by declining to upgrade.
Upgrade Unsupported	Upgrade of the IP variation is not supported in the current version of the Quartus II software due to IP core end of life or incompatibility with the current version of the Quartus II software. You are prompted to replace the obsolete IP core with a current equivalent IP core from the IP Catalog.

Before you begin

- Archive the Quartus II project containing outdated IP cores in the original version of the Quartus II software: Click **Project** > **Archive Project** to save the project in your previous version of the Quartus II software. This archive preserves your original design source and project files.
- Restore the archived project in the latest version of the Quartus II software: Click **Project** > **Restore Archived Project**. Click **OK** if prompted to change to a supported device or overwrite the project database. File paths in the archive must be relative to the project directory. File paths in the archive must reference the IP variation **.v** or **.vhd** file or **.qsys** file (not the **.qip** file).

1. In the latest version of the Quartus II software, open the Quartus II project containing an outdated IP core variation. The **Upgrade IP Components** dialog automatically displays the status of IP cores in your project, along with instructions for upgrading each core. Click **Project** > **Upgrade IP Components** to access this dialog box manually.
2. To simultaneously upgrade all IP cores that support automatic upgrade, click **Perform Automatic Upgrade**. The **Status** and **Version** columns update when upgrade is complete. Example designs provided with any Altera IP core regenerate automatically whenever you upgrade the IP core.

Figure 26: Upgrading IP Cores



Example 1: Upgrading IP Cores at the Command Line

You can upgrade IP cores that support auto upgrade at the command line. IP cores that do not support automatic upgrade do not support command line upgrade.

- To upgrade a single IP core that supports auto-upgrade, type the following command:

```
quartus_sh –ip_upgrade –variation_files <my_ip_filepath/my_ip>.<hdl>
<qii_project>

Example:
quartus_sh -ip_upgrade -variation_files mega/pll25.v hps_testx
```

- To simultaneously upgrade multiple IP cores that support auto-upgrade, type the following command:

```
quartus_sh –ip_upgrade –variation_files "<my_ip_filepath/my_ip1>.<hdl>;
<my_ip_filepath/my_ip2>.<hdl>"  <qii_project>

Example:
quartus_sh -ip_upgrade -variation_files "mega/pll_tx2.v;mega/pll3.v" hps_testx
```

Note: IP cores older than Quartus II software version 12.0 do not support upgrade. Altera verifies that the current version of the Quartus II software compiles the previous version of each IP core. The *Altera IP Release Notes* reports any verification exceptions for Altera IP cores. Altera does not verify compilation for IP cores older than the previous two releases.

Related Information

Altera IP Release Notes

Migrating IP Cores to a Different Device

IP migration allows you to target the latest device families with IP originally generated for a different device. Some Altera IP cores require individual migration to upgrade. The **Upgrade IP Components** dialog box prompts you to double-click IP cores that require individual migration.

1. To display IP cores requiring migration, click **Project** > **Upgrade IP Components**. The **Description** field prompts you to double-click IP cores that require individual migration.
2. Double-click the IP core name, and then click **OK** after reading the information panel.
   The parameter editor appears showing the original IP core parameters.
3. For the **Currently selected device family**, turn off **Match project/default**, and then select the new target device family.
4. Click **Finish**, and then click **Finish** again to migrate the IP variation using best-effort mapping to new parameters and settings. Click **OK** if you are prompted that the IP core is unsupported for the current device. A new parameter editor opens displaying best-effort mapped parameters.
5. Click **Generate HDL**, and then confirm the **Synthesis** and **Simulation** file options. Verilog is the parameter editor default HDL for synthesis files. If your original IP core was generated for VHDL, select **VHDL** to retain the original output HDL format.
6. To regenerate the new IP variation for the new target device, click **Generate**. When generation is complete, click **Close**.
7. Click **Finish** to complete migration of the IP core. Click **OK** if you are prompted to overwrite IP core files. The **Device Family** column displays the migrated device support. The migration process replaces <*my_ip*>**.qip** with the <*my_ip*>**.qsys** top-level IP file in your project.

   Note: If migration does not replace <*my_ip*>**.qip** with <*my_ip*>**.qsys**, click **Project > Add/Remove Files in Project** to replace the file in your project.

8. Review the latest parameters in the parameter editor or generated HDL for correctness. IP migration may change ports, parameters, or functionality of the IP core. During migration, the IP core's HDL generates into a library that is different from the original output location of the IP core. Update any assignments that reference outdated locations. If your upgraded IP core is represented by a symbol in a supporting Block Design File schematic, replace the symbol with the newly generated <*my_ip*>**.bsf** after migration.

   Note: The migration process may change the IP variation interface, parameters, and functionality. This may require you to change your design or to re-parameterize your variant after the **Upgrade IP Components** dialog box indicates that migration is complete. The **Description** field identifies IP cores that require design or parameter changes.

Related Information

Altera IP Release Notes

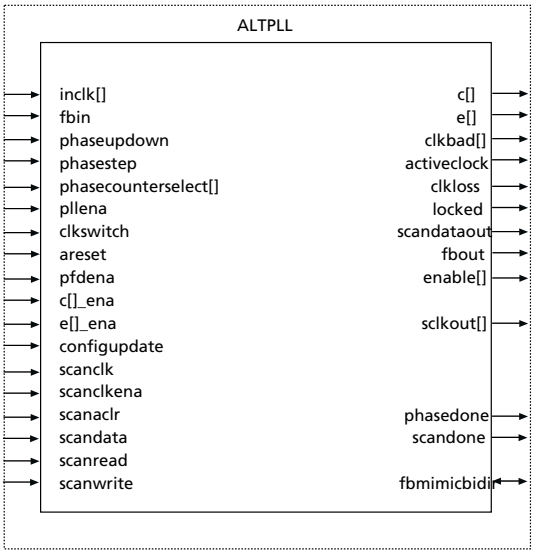
Ports and Parameters

The ALTPLL IP core ports and parameters are available to customize the ALTPLL IP core according to your application.

The options you select in the parameter editor affect the ports and parameters settings of the IP core. Most of these options are device-dependent, and some options depend on one another. The options you select enable the PLL features, and some might generate top-level ports in the ALTPLL IP core.

The following figure shows the top-level ports of the ALTPLL IP core.

Figure 27: Input and Output Ports



The parameters are only relevant if you bypass the parameter editor GUI and instantiate the IP core directly in your design. The details of these parameters are hidden from the parameter editor.

ALTPLL Input Ports

Table 16: ALTPLL Input Ports

Port Name[6]	Condition	Description
areset	Optional	Resets all counters to initial values, including the GATE_LOCK_COUNTER parameter.
c[]_ena	Optional	An enable input port for the output clock, c[]. Available for Stratix and Stratix GX devices only.

[6] Replace brackets, [], in the port name with integer to get the exact name. For example, inclk0, inclk1, c1_ena, and e0_ena.

Port Name[6]	Condition	Description
clkswitch	Optional	The control input port to dynamically toggle between clock input ports (inclk0 and inclk1 ports), or to manually override the automatic clock switchover. You should create the clkswitch port if only the inclk1 port is created.
configupdate	Optional	Dynamic full PLL reconfiguration.
e[]_ena[]	Optional	An enable input port for the external output clock, e[]. Available for Stratix and Stratix GX devices only.
fbin	Optional	The external feedback input port for the PLL. This port must be created if the PLL operates in the external feedback mode. To complete the feedback loop, there must be a board-level connection between the fbin port and the external clock output port of the PLL. In the Stratix III device, if the PLL operates in the zero-delay buffer mode and the fbmimicbidir port is not used, you must connect the fbin port to the fbout port.
inclk[]	Required	The clock inputs that drive the clock network. If more than one inclk[] port is created, you must use the clkselect port to specify which clock is used. The inclk0 port must always be connected; connect other clock inputs if switching is necessary. A dedicated clock pin or PLL output clock can drive this port.

[6] Replace brackets, [], in the port name with integer to get the exact name. For example, inclk0, inclk1, c1_ena, and e0_ena.

Port Name[6]	Condition	Description
pfdena	Optional	Enables the phase frequency detector (PFD). When the PFD is disabled, the PLL continues to operate regardless of the input clock. Because the PLL output clock frequency does not change for some time, you can use the pfdena port as a shutdown or cleanup function when a reliable input clock is no longer available.
phasecounterselect[]	Optional	Specifies counter select. Only available for Arria II GX, Stratix III, and Cyclone III devices onwards.
phasestep	Optional	Specifies dynamic phase shifting. Only available for Arria II GX, Stratix III, and Cyclone III devices onwards.
phaseupdown	Optional	Specifies dynamic phase adjustment up or down. Only available for Arria II GX, Stratix III, and Cyclone III devices onwards.
pllena	Optional	The PLL enable port. When the pllena port is asserted, the PLL drives out a signal, and vice versa. When the pllena port is re-asserted, the PLL has to re-lock. All PLLs on the device share the same pllena port.
scanaclr	Optional	Asynchronous clear port for the real-time programming scan chain, or the serial scan chain. Only available for Stratix and Stratix GX devices.
scanclk	Optional	Input clock port for the serial scan chain. Not available for Cyclone and Cyclone II devices.

(6) Replace brackets, [], in the port name with integer to get the exact name. For example, inclk0, inclk1, c1_ena, and e0_ena.

Port Name[6]	Condition	Description
scanclkena	Optional	Clock enable port for the serial scan chain. Available only for Arria II GX, Cyclone III, HardCopy III, and Stratix III devices onwards.
scandata	Optional	Contains the data for the serial scan chain. Not available for Cyclone and Cyclone II devices.
scanread	Optional	Port that controls the serial scan chain to read input from the scandata port. Only available for Arria GX, HardCopy II, Stratix II, and Stratix II GX devices.
scanwrite	Optional	Port that controls the real-time programming scan chain to write to the PLL. Only available for Arria GX, HardCopy II, Stratix II, and Stratix II GX devices.

ALTPLL Output Ports

Table 17: ALTPLL Output Ports

Port Name [7]	Condition	Description
activelock	Optional	Specifies which clock is used as the primary reference clock when the clock switchover circuit initiates. If the inclk0 is in use, the activeclock port goes low. If the inclk1 is in use, the activeclock port goes high. You can set the PLL to automatically initiate the clock switchover when the primary reference clock is not toggling correctly, or you can manually initiate the clock switchover using the clkswitch input port.
c[]	Required	The clock output of the PLL.

[6] Replace brackets, [], in the port name with integer to get the exact name. For example, inclk0, inclk1, c1_ena, and e0_ena.

[7] Replace the brackets, [], in the port name with an integer to get the exact name (for example, c0, c1, c3, e0, e1, e2, enable1 and sclkout0).

Port Name [7]	Condition	Description
clkbad[]	Optional	clkbad1 and clkbad0 ports check for input clock toggling. If the inclk0 port stops toggling, the clkbad0 port goes high. If the inclk1 port stops toggling, the clkbad1 port goes high.
clkloss	Optional	This output port acts as an indicator when the clock switchover circuit initiates. Only available for Arria GX, HardCopy II, Stratix, Stratix GX, Stratix II, and Stratix II GX devices.
enable[]	Optional	Enable pulse output port. This port is available only when the PLL is in LVDS mode. Only available for Arria GX, HardCopy II, Stratix II, and Stratix II GX devices.
e[]	Optional	The clock output of the PLL that feeds the dedicated clock pins on the device. Only available for Cyclone, Stratix, and Stratix GX devices.

[7] Replace the brackets, [], in the port name with an integer to get the exact name (for example, c0, c1, c3, e0, e1, e2, enable1 and sclkout0).

Port Name [7]	Condition	Description
fbout	Optional	The port that feeds the fbin port through the mimic circuitry. If a feedback path is not connected, the compiler automatically connects the fbout port directly to the fbin port. Additionally, a clkbuf primitive is added to specify the resource type used, similarly to other clock networks. The fbout port is available only if the PLL is in external feedback mode. For Stratix III and Stratix IV device families, if the PLL operates in zero-delay buffer mode, and the fbmimicbidir port is not used, you must perform the following steps: **1.** Replace the brackets, [], in the port name with an integer to get the exact name (for example, c0, c1, c3, e0, e1, e2, enable1 and sclkout0). **2.** Instantiate an ALT_IOBUF IP core. **3.** Connect the fbout and fbin ports to the o and i ports of the ALT_IOBUF instantiation, respectively. **4.** Connect the bidirectional port of the ALT_IOBUF instantiation to a bidirectional pin, and set the oe port of the ALT_IOBUF instantiation to 1.

[7] Replace the brackets, [], in the port name with an integer to get the exact name (for example, c0, c1, c3, e0, e1, e2, enable1 and sclkout0).

Port Name [7]	Condition	Description
locked	Optional	This output port acts as an indicator when the PLL has reached phase-locked. The locked port stays high as long as the PLL is locked, and stays low when the PLL is out-of-lock. The number of cycles needed to gate the locked signal is based on the PLL input clock. The gated-lock circuitry is clocked by the PLL input clock. The maximum lock time for the PLL is provided in the *DC and Switching Characteristics* chapter of the device handbook. Take the maximum lock time of the PLL and divide it by the period of the PLL input clock. The result is the number of clock cycles needed to gate the locked signal. The lock signal is an asynchronous output of the PLL. The PLL lock signal is derived from the reference clock and feedback clock feeding the Phase Frequency Detector (PFD). Reference clock = Input Clock/N Feedback clock = VCO/M The PLL asserts the locked port when the phases and 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 goes beyond the lock circuit tolerance, the PLL loses lock.
phasedone	Optional	This output port indicates that dynamic phase reconfiguration is completed. Only available for Arria II GX, Cyclone III, HardCopy III onwards, and Stratix III onwards.

[7] Replace the brackets, [], in the port name with an integer to get the exact name (for example, c0, c1, c3, e0, e1, e2, enable1 and sclkout0).

Port Name [7]	Condition	Description
scandataout	Optional	The data output for the serial scan chain. You can use the scandataout port to determine when PLL reconfiguration is completed. The last output is cleared when reconfiguration completes. Not available for Cyclone and Cyclone II devices.
scandone	Optional	This output port indicates that the scan chain write operation is initiated. The scandone port goes high when the scan chain write operation initiates, and goes low when the scan chain write operation completes. Not available for Cyclone, Cyclone II, Stratix, and Stratix GX devices.
sclkout[]	Optional	Serial clock output port. This port is available only when the PLL is in LVDS mode. Only available for Arria GX, HardCopy II, Stratix II, and Stratix II GX devices.
vcounverrange	Optional	This output port indicates that the VCO frequency has exceeded the legal VCO range.
vcounderrange	Optional	This output port indicates that the VCO frequency has not met the legal VCO range.

[7] Replace the brackets, [], in the port name with an integer to get the exact name (for example, c0, c1, c3, e0, e1, e2, enable1 and sclkout0).

ALTPLL Bidirectional Port

Table 18: ALTPLL Bidirectional Output Port

Port Name	Condition	Description
fbmimicbidir	Optional	The bidirectional port that connects to the mimic circuitry. This port is available only for Stratix III and Stratix IV device families, and only when the PLL is in zero-delay buffer mode. This port must be connected to a bidirectional pin that is placed on the positive feedback dedicated output pin of the PLL.

ALTPLL Parameters

In the ALTPLL parameters, the c[] and e[] ports are named CLK[] and EXTCLK[], respectively. This is to differentiate them from the parameters used to describe the *C* and *E* counters of the PLL.

Replace the brackets, [], in the parameter name with an integer to get the exact parameter name. For example, the C[]_HIGH parameter can have up to 10 variations as described in the parameter description. The variations are C0_HIGH, C1_HIGH, C2_HIGH, C3_HIGH, C4_HIGH, C5_HIGH, C6_HIGH, C7_HIGH, C8_HIGH, and C9_HIGH.

A string must be contained within a pair or double quotes. For example, to specify a medium bandwidth type of PLL, BANDWIDTH_TYPE="MEDIUM".

An integer must not be contained within a pair or double quotes. For example, to specify a 40% duty cycle for the output clock port, c5, CLK5_DUTY_CYCLE=40.

Table 19: ALTPLL Parameters

Parameter	Type	Description
BANDWIDTH	String	Specifies the bandwidth value of the PLL in MHz. If this parameter is not specified, the Compiler automatically determines the value of the BANDWIDTH parameter to satisfy other PLL settings.
BANDWIDTH_TYPE	String	Specifies the type of bandwidth for the BANDWIDTH parameter. Values are AUTO, LOW, MEDIUM, or HIGH. If omitted, the default value is AUTO. For the low bandwidth option, the PLL has a better jitter rejection but slower lock time. For the high bandwidth option, the PLL has a faster lock time but tracks more jitter. The medium option is a balance between the other two options.
C[]_HIGH	Integer	Specifies the value of the high period count for the corresponding counter, C[9..0]. If omitted, default is 1. Counters C[9..5] are not available in Cyclone III devices onwards.

Parameter	Type	Description
C[]_INITIAL	Integer	Specifies the initial value for the corresponding counter, C[9..0]. If omitted, default is 1. Counters C[9..5] are not available in Cyclone III devices onwards.
C[]_LOW	Integer	Specifies the value of the low period count for the corresponding counter, C[9..0]. If omitted, default is 1. Counters C[9..5] are not available in Cyclone III devices onwards.
C[]_MODE	String	Specifies the operation mode for the counter, C[9..0]. The values are BYPASS, ODD, or EVEN. If omitted, the default is BYPASS. Counters C[9..5] are not available in Cyclone III devices onwards.
C[]_PH	Integer	Specifies the phase tap for the counter, C[9..0]. If omitted, default is 0. Counters C[9..5] are not available in Cyclone III devices onwards.
C[]_TEST_SOURCE	Integer	Specifies the test source for the counter, C[9..0]. If omitted, default is 0. Counters C[9..5] are not available in Cyclone III devices onwards.
C[]_USE_CASC_IN	String	Specifies whether to use cascade input for the counter, C[9..1]. Values are ON or OFF. If omitted, default is OFF. Counters C[9..5] are not available in Cyclone III devices onwards.
CHARGE_PUMP_CURRENT	Integer	Specifies the value of the charge pump current in microamperes (mA).
CLK[]_COUNTER	String	Specifies the counter for the corresponding output clock port, CLK[9..0]. Values are UNUSED, C0, C1, C2, C3, C4, C5, C6, C7, C8 or C9. If omitted, the default is C0. This parameter is not available for Cyclone II and Stratix II devices. Counters CLK[9..5]_COUNTER are not available for Cyclone III devices onwards.
CLK[]_DIVIDE_BY	Integer	Specifies the integer division factor for the VCO frequency of the corresponding output clock port, CLK[9..0] port. The parameter value must be greater than 0. Specify this parameter only if the corresponding CLK[9..0] port is used; however, it is not required if a Clock Settings assignment is specified for the corresponding CLK[9..0] port. If omitted, the default is 0. Parameters CLK[9..5]_DIVIDE_BY are not available in Cyclone III devices.
CLK[]_DUTY_CYCLE	Integer	Specifies the duty cycle in percentage for the corresponding output clock port, clk[9..0]. If omitted, the default is 50. Parameters CLK[9..5]_DUTY_CYCLE are not available in Cyclone III devices.

Parameter	Type	Description
CLK[]_MULTIPLY_BY	Integer	Specifies the integer multiplication factor for the VCO frequency for the corresponding output clock port, CLK[9..0]. The parameter value must be greater than 0. Specify this parameter only if the corresponding CLK[9..0] port is used; however, it is not required if a Clock Settings assignment is specified for the corresponding CLK[9..0] port. If omitted, the default is 0. Parameters CLK[9..5]_MULTIPLY_BY are not available in Cyclone III devices.
CLK[]_OUTPUT_FREQUENCY	Integer	Specifies the output frequency of the corresponding output clock port, CLK[9..0].This parameter is ignored if the corresponding CLK[9..0] port is not used. This parameter is unavailable if multiplication or division factors are specified. If omitted, the default is 0.
CLK[]_PHASE_SHIFT	String	Specifies the phase shift for the corresponding output clock port, CLK[9..0], in picoseconds (ps). If omitted, the default is 0. Parameters CLK[9..5]_PHASE_SHIFT are not available in Cyclone III devices.
CLK[]_TIME_DELAY	String	Specifies, in picoseconds (ps), the delay value to be applied to the corresponding output clock port, CLK[9..0]. This parameter affects only the corresponding CLK[9..0] port and is independent of the corresponding CLK[9..0]_PHASE_SHIFT parameter; therefore, both parameters can be used simultaneously. If no units are specified, default is picoseconds (ps). Legal time delay values range from -3 ns to 6 ns in increments of 0.25 ns. These values should not be used as parameters except when reprogramming the PLL using the real-time programming interface.
CLK[]_USE_EVEN_COUNTER_MODE	String	Specifies whether the clock output needs to be forced to use even counter mode for the corresponding CLK[9..0] port. Values are ON or OFF. If omitted, the default is OFF.
CLK[]_USE_EVEN_ COUNTER_VALUE	String	Specifies whether the clock output needs to be forced to use even counter values for the corresponding CLK[9..0] port. Values are ON or OFF. If omitted, the default is OFF.

Parameter	Type	Description
COMPENSATE_CLOCK	String	Specifies the output clock port which should be compensated for. • If the OPERATION_MODE parameter is set to normal mode, the values are CLK[], GCLK[], LCLK[] , or LVDSCLK[]. Default is CLK0. • If the OPERATION_MODE parameter is set to zero-delay buffer mode, value is EXTCLK[]. Default is EXTCLK0. • If the OPERATION_MODE parameter is set to source-synchronous mode, the values are CLK[], LCLK[], GCLK[], or LVDSCLK[]. This clock cannot be offset with respect to the reference clock. This relationship is preserved closely even upon temperature and frequency changes.
DOWN_SPREAD	String	Specifies the down spectrum percentage. Values range from 0 through 0.5.
E[]_HIGH	Integer	Specifies the high period count for the corresponding E[3..0] counter. Values range from 1 through 512. If omitted, the default is 1.
E[]_INITIAL	Integer	Specifies the initial value for the corresponding E[3..0] counter. Values range from 1 through 512. If omitted, the default is 1.
E[]_LOW	Integer	Specifies the low period count for the corresponding E[3..0] counter. Values range from 1 through 512. If omitted, the default is 1.
E[]_MODE	String	Specifies the mode for the corresponding E[3..0] counter. Values are BYPASS, ODD, or EVEN. If omitted, the default is BYPASS.
E[]_PH	Integer	Specifies the phase tap for the corresponding E[3..0] counter. Values range from 0 through 7. If omitted, the default is 0.
E[]_TIME_DELAY	String	Specifies, in nanoseconds (ns), the time delay for the corresponding E[3..0] counter. Values range from 0 ns to 3 ns. If omitted, the default is 0.
ENABLE[]_COUNTER	String	Specifies the counter for the corresponding ENABLE[1..0] port. Values are L0 or L1.
ENABLE_SWITCH_OVER_COUNTER	String	Specifies whether to use the switchover counter. Values are ON or OFF. If omitted, the value is OFF.

Parameter	Type	Description
EXTCLK[]_COUNTER	String	Specifies the external counter for the corresponding external clock output port, EXTCLK[3..0]. Values are E0, E1, E2, or E3. If omitted, the default is E0. This parameter is only available for Stratix, Stratix GX, and Cyclone (EXTCLK0) devices.
EXTCLK[]_DIVIDE_BY	Integer	Specifies the integer division factor for the corresponding external clock output port, EXTCLK[3..0], with respect to the input clock frequency. The parameter value must be greater than 0. You can specify this parameter only if the corresponding EXTCLK[3..0] port is used; however, it is not required if a Clock Settings assignment is specified for the corresponding EXTCLK[3..0] port. If omitted, the default is 1. This parameter is not available for Stratix II devices.
EXTCLK[]_DUTY_CYCLE	Integer	Specifies the duty cycle in percentage for the corresponding external clock output port, EXTCLK[3..0]. If omitted, the default is 50. This parameter is not available for Stratix II devices.
EXTCLK[]_MULTIPLY_BY	Integer	Specifies the integer multiplication factor for the corresponding external clock output port, EXTCLK[3..0], with respect to the input clock frequency. The parameter value must be greater than 0. You can specify this parameter only if you use the corresponding EXTCLK[3..0] port. However, it is not required if a Clock Settings assignment is specified for the corresponding EXTCLK[3..0] port. If omitted, the default is 1. This parameter is not available for Stratix II devices.
EXTCLK[]_PHASE_SHIFT	String	Specifies the phase shift for the corresponding external clock output port, EXTCLK[3..0]. This parameter is not available for Stratix II devices.
EXTCLK[]_TIME_DELAY	String	Specifies, in picoseconds (ps), a delay value to be applied to the corresponding external clock output port, EXTCLK[3..0]. This parameter affects only the corresponding EXTCLK[3..0] port and is independent of the EXTCLK[3..0]_PHASE_SHIFT parameter; therefore you can use both parameters simultaneously. If no units are specified, picoseconds (ps) are assumed. Legal values range from –3 ns to 6 ns in increments of 0.25 ns. Do not use these values as parameters except when reprogramming the PLL using the real-time programming interface. This parameter is not available for Stratix II devices.

Parameter	Type	Description
FEEDBACK_SOURCE	String	Specifies which clock output has a board-level connection to the fbin port. If the OPERATION_MODE parameter is set to EXTERNAL_FEEDBACK, the FEEDBACK_SOURCE parameter is needed. Values are EXTCLK[]. If omitted, the value is EXTCLK0.
G[]_HIGH	Integer	Specifies the high period count for the corresponding G[3..0] counter. Values range from 1 to 512. If omitted, the default is 1.
G[]_INITIAL	Integer	Specifies the initial value for the corresponding G[3..0] counter. Values range from 1 to 512. If omitted, the default is 1.
G[]_LOW	Integer	Specifies the low period count for the corresponding G[3..0] counter. Values range from 1 to 512. If omitted, the default is 1.
G[]_MODE	String	Specifies the mode for the corresponding G[3..0] counter. Values are BYPASS, ODD, or EVEN. If omitted, the default is BYPASS.
G[]_PH	Integer	Specifies the phase tap for the corresponding G[3..0] counter. Values range from 0 to 7. If omitted, the default is 0.
G[]_TIME_DELAY	String	Specifies, in nanoseconds (ns), the time delay for the corresponding G[3..0] counter. Values range from 0 ns to 3 ns. If omitted, the default is 0.
GATE_LOCK_COUNTER	Integer	Specifies the value for the 20-bit counter that gates the locked output port before sending it to the locked port. This parameter is required for simulation in third-party simulators.
GATE_LOCK_SIGNAL	String	Specifies whether the locked port should be gated internally with a 20-bit programmable counter so it does not oscillate during initial power-up. Values are NO or YES. If omitted, default is NO.
INCLK[]_INPUT_FREQUENCY	Integer	Specifies the input frequency for the input clock port, inclk[1..0]. The compiler uses the frequency of the clk0 port to calculate the PLL parameters, but also analyzes and reports the phase shifts for the clk1 port.
INTENDED_DEVICE_FAMILY	String	This parameter is used for modeling and behavioral simulation purposes. The default is NONE
INVALID_LOCK_MULTIPLIER	Integer	Specifies the scaling factor, in half-clock cycles, for which the clock output ports must be out-of-lock before the locked pin goes low.

Parameter	Type	Description
L[]_HIGH	Integer	Specifies the high period count for the corresponding L[1..0] counter. Values range from 1 to 512. If omitted, the default is 1.
L[]_INITIAL	Integer	Specifies the initial value for the corresponding L[1..0] counter. Values range from 1 to 512. If omitted, the default is 1.
L[]_LOW	Integer	Specifies the low period count for the corresponding L[1..0] counter. Values range from 1 to 512. If omitted, the default is 1.
L[]_MODE	String	Specifies the mode for the corresponding L[1..0] counter. Values are BYPASS, ODD or EVEN. If omitted, the default is BYPASS.
L[]_PH	Integer	Specifies the phase tap for the corresponding L[1..0] counter. Values range from 0 to 7. If omitted, the default is 0.
L[]_TIME_DELAY	String	Specifies, in nanoseconds (ns), the time delay for the corresponding L[1..0] counter. Values range from 0 ns to 3 ns. If omitted, the default is 0.
LOCK_HIGH	Integer	Specifies the number of half-clock cycles that the output clocks must be locked before the locked port goes high. This parameter is required for simulation in third-party simulators. Available for Stratix IV, Stratix III, Cyclone IV, and Cyclone III devices only.
LOCK_LOW	Integer	Specifies the number of half-clock cycles that the output clocks must be out-of-lock before the locked port goes low. This parameter is required for simulation in third-party simulators. Available for Stratix IV, Stratix III, Cyclone IV, and Cyclone III devices only.
LOCK_WINDOW_UI	String	Specifies the value of the LOCK_WINDOW_UI setting. If omitted, default is 0.05.
LOOP_FILTER_C	String	Specifies, in picofarads (pF), the value of the loop capacitor. Values range from 5 to 20 pF. The Compiler cannot achieve all values. If omitted, the default value is 10.
LOOP_FILTER_R	String	Specifies, in kilo ohms (K Ohm), the value of the loop resistor. Values range from 1 K to 20 K. The Compiler cannot achieve all values.
LPM_HINT	String	Allows you to specify Altera-specific parameters in VHDL Design Files (**.vhd**). The default is UNUSED.
LPM_TYPE	String	Identifies the library of parameterized modules (LPM) entity name in VHDL Design Files.

Parameter	Type	Description
M	Integer	Specifies the modulus for the M counter. Provides direct access to the internal PLL parameters. If the M parameter is specified, all advanced parameters must be used. Values range from 1 to 512. If omitted, the default is 0.
M_INITIAL	Integer	Specifies the initial value for the M counter. Provides direct access to the internal PLL parameters. If the M_INITIAL parameter is specified, all advanced parameters must be used. Values range from 0 to 512. If omitted, the default is 0.
M_PH	Integer	Specifies the phase tap for the M counter. Values range from 0 to 7. If omitted, the default is 0.
M_TEST_SOURCE	Integer	Specifies the test source for the M counter. If omitted, default is 5.
M_TIME_DELAY	Integer	Specifies, in nanoseconds (ns), the time delay for the M counter. Values range from 0 ns through 3 ns. If omitted, the default is 0. This parameter is not available for Cyclone II and Stratix II devices.
M2	Integer	Specifies the spread spectrum modulus for the M counter. Provides direct access to the internal PLL parameters. If the M2 parameter is specified, all advanced parameters must be used. Values range from 1 to 512. If omitted, the value is 1.
N	Integer	Specifies the modulus for the N counter. Provides direct access to the internal PLL parameters. If the N parameter is specified, all advanced parameters must be used. Values range from 1 to 512. If omitted, the default is 1.
N_TIME_DELAY	Integer	Specifies, in nanoseconds (ns), the time delay for the N counter. Values range from 0 ns through 3 ns. If omitted, the default is 0. This parameter is not available for Cyclone II and Stratix II devices.
N2	Integer	Specifies the spread spectrum modulus for the N counter. Provides direct access to the internal PLL parameters. If the N2 parameter is specified, all advanced parameters must be used. Values range from 1 through 512. If omitted, the default is 1.

Parameter	Type	Description
OPERATION_MODE	String	

Parameter	Type	Description
		Specifies the operation of the PLL. Values are EXTERNAL_FEEDBACK, NO_COMPENSATION, NORMAL, ZERO_DELAY_BUFFER, and SOURCE_SYNCHRONOUS. If omitted, the default is NORMAL. • In no compensation mode, the PLL does not align a clock to the input, which leads to better jitter performance. • In source-synchronous mode, the clock delay from pin to I/O input register matches the data delay from pin to I/O input register. • Source-synchronous mode allows the clock delay from pin to I/O input register to match the data delay from pin to I/O input register. • In normal mode, the PLL compensates for the delay of the internal clock network used by the clock output specified in the COMPENSATE_CLOCK parameter. If the PLL is also used to drive an external clock output pin, a corresponding phase shift of the output pin results. • In 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. If the PLL is also used to drive the internal clock network, a corresponding phase shift of that network results. • In external feedback mode, the fbin input port must be connected to an input pin, and there must be a board-level connection between this input pin and an external clock output pin, which is specified with FEEDBACK_SOURCE parameter. The fbin port is aligned with the input clock. Use the maximum input delay assignment on the fbin port to specify external board delay. Note that for source-synchronous mode and zero-delay buffer mode, you need to make assignments (in this case, the PLL_COMPENSATE assignment) in addition to setting the appropriate mode in the IP core. This allows you to specify an output pin as a compensation target for a PLL in zero-delay buffer or external feedback mode, or an input pin or a group of input pins as compensation targets for a PLL in Source- Synchronous mode. If assigned to an output pin, the pin must be fed by the external clock output port of a PLL in a Stratix, HardCopy Stratix or Cyclone device, or the compensated clock output port of a PLL in other devices. Any other output pins fed by the same PLL generally are not delay compensated, especially if they have different I/O standards. If assigned to an input pin or a group of input pins, the input pins must drive input registers that are clocked by

Parameter	Type	Description
		the compensated clock output port of a PLL in source-synchronous mode. This parameter is ignored if it is applied to anything other than an output or input pin as described above
PFD_MAX	Integer	Specifies the maximum value for the PFD pin. If omitted, the default is 0.
PFD_MIN	Integer	Specifies the minimum value for the PFD pin. If omitted, the default is 0.
PLL_TYPE	String	Specifies the type of PLL to instantiate. Values are AUTO, ENHANCED, FAST, TOP_BOTTOM and LEFT_RIGHT. If omitted, the default is AUTO.
PORT_ACTIVECLOCK	String	Specifies the ACTIVECLOCK port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_ARESET	String	Specifies the ARESET port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_CLK[]	String	Specifies the CLK[9..0] port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_CLKBAD[]	String	Specifies the CLKBAD[1..0] port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_CLKENA[]	String	Specifies the CLKENA[5..0] port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_CLKLOSS	String	Specifies the CLKLOSS port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_CLKSWITCH	String	Specifies the CLKSWITCH port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_CONFIGUPDATE	String	Specifies the CONFIGUPDATE port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_ENABLE[]	String	Specifies the ENABLE[1..0] port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.

Parameter	Type	Description
PORT_EXTCLK[]	String	Specifies the EXTCLK[3..0] port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_EXTCLKENA[]	String	Specifies the EXTCLKENA[3..0] port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_FBIN	String	Specifies the FBIN port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_FBOUT	String	Specifies the FBOUT port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_INCLK[]	String	Specifies the INCLK[1..0] port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_LOCKED	String	Specifies the LOCKED port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_PFDENA	String	Specifies the PFDENA port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_PHASECOUNTERSELECT	String	Specifies the PHASECOUNTERSELECT port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_PHASEDONE	String	Specifies the PHASEDONE port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_PHASESTEP	String	Specifies the PHASESTEP port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_PHASEUPDOWN	String	Specifies the PHASEUPDOWN port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_PLLENA	String	Specifies the PLLENA port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_SCANACLR	String	Specifies the SCANACLR port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.

Parameter	Type	Description
PORT_SCANCLK	String	Specifies the SCANCLK port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_SCANCLKENA	String	Specifies the SCANCLKENA port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_SCANDATA	String	Specifies the SCANDATA port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_SCANDONE	String	Specifies the SCANDONE port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_SCANREAD	String	Specifies the SCANREAD port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_SCANWRITE	String	Specifies the SCANWRITE port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_SCLKOUT[]	String	Specifies the SCLKOUT[1..0] port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_VCOOVERRANGE	String	Specifies the VCOOVERRANGE port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PORT_VCOUNDERRANGE	String	Specifies the VCOUNDERRANGE port connectivity. Values are PORT_USED or PORT_UNUSED. If omitted, the default is PORT_UNUSED.
PRIMARY_CLOCK	String	Specifies the primary reference clock of the PLL. Values are INCLK0 or INCLK1. If omitted, the default is INCLK0. Use the clock switch scheme to switch between clocks. Automatic clock switchover is defined differently in different device families.
QUALIFY_CONF_DONE	String	The values are ON or OFF. If omitted, the default is OFF.
SCAN_CHAIN	String	Specifies the length of the scan chain. Values are LONG or SHORT. If omitted, the default is LONG. If set to LONG, the scan chain length is 10 counters. If set to SHORT, the scan chain length is 6 counters.

Parameter	Type	Description
SCLKOUT[]_PHASE_SHIFT	String	Specifies, in picoseconds (ps), the phase shift for the corresponding sclkout[1..0] port. The maximum phase value is 7/8 of one VCO period. The VCO phase tap is shared with the corresponding clock output port, clk[1..0], and must have the same phase amount that is less than one VCO period. In LVDS mode, this parameter default value is 0.
SELF_RESET_ON_GATED_LOSS_LOCK	String	Specifies if the self-reset on gated-lock-loss feature is used. Values are ON or OFF. If omitted, the default is OFF.
SELF_RESET_ON_LOSS_LOCK	String	Specifies if the self-reset on loss lock feature is used. Values are ON or OFF. If omitted, the default is OFF.
SKIP_VCO	Integer	Specifies if the VCO is skipped. Values are ON or OFF. If omitted, the default is OFF.
SPREAD_FREQUENCY	Integer	Specifies, in picoseconds (ps), the modulation frequency for spread spectrum. If omitted, the default is 0.
SS	Integer	Specifies the modulus for the spread spectrum counter. Provides direct access to the internal PLL parameters. If the SS parameter is specified, all advanced parameters must be used. Values range from 1 to 32768. If omitted, the default is 1.
SWITCH_OVER_COUNTER	Integer	Specifies the number of clock cycles to switch the input clock after the switchover circuit is initiated. Values are 0 through 31. If omitted, the default is 0.
SWITCH_OVER_ON_GATED_LOCK	String	Specifies whether the gated-lock condition should initiate a clock switch over. Values are ON or OFF. If omitted, the value is OFF.
SWITCH_OVER_ON_LOSSCLK	String	Specifies whether the loss-of-lock condition should initiate a clock switch over. Values are ON or OFF. If omitted, the value is OFF.
SWITCH_OVER_TYPE	String	Specifies the switchover type. Values are AUTO or MANUAL. If omitted, the default is AUTO.

Parameter	Type	Description
USING_FBMIMICBIDIR_PORT	String	Specifies whether the fbmimicbidir port is used. Values are ON or OFF. If omitted, the default is OFF. For Stratix III and Stratix IV device families, with the OPERATION_MODE parameter set to ZERO_DELAY_BUFFER, and the USING_FBMIMICBIDIR_PORT parameter is set to OFF, you must perform the following steps: 1. Instantiate an ALT_IOBUF IP core. 2. Connect the fbout and fbin ports to the o and i ports of the ALT_IOBUF instantiation, respectively. 3. Connect the bidirectional port of the ALT_IOBUF instantiation to a bidirectional pin, and set the oe port of the ALT_IOBUF instantiation to 1. If the OPERATION_MODE parameter is set to ZERO_DELAY_BUFFER, and the USING_FBMIMICBIDIR_PORT parameter is set to ON, connect the fbmimicbidir port to a bidirectional pin. This pin must be placed on the positive feedback dedicated output pin of the PLL.
VALID_LOCK_MULTIPLIER	Integer	Specifies the scaling factor, in half-clock cycles, for which the clock output ports must be locked before the locked pin goes high.
VCO_CENTER	Integer	Specifies the center value for the VCO pin. Use for simulation purposes only.
VCO_DIVIDE_BY	Integer	Specifies the integer division factor for the VCO pin. If omitted, the default is 0. If VCO_FREQUENCY_CONTROL is set to MANUAL_PHASE, specify the VCO frequency as a phase shift step value; that is, one-eighth of the VCO period.
VCO_FREQUENCY_CONTROL	String	Specifies the integer division factor for the VCO pin. If omitted, the default is 0. If VCO_FREQUENCY_CONTROL is set to MANUAL_PHASE, specify the VCO frequency as a phase shift step value; that is, one-eighth of the VCO period. Specifies the frequency control value for the VCO pin. Values are AUTO, MANUAL_FREQUENCY, and MANUAL_PHASE. If omitted, the default is AUTO. • AUTO—VCO_MULTIPLY_BY and VCO_DIVIDE_BY values are ignored and VCO frequency is set automatically. • MANUAL_FREQUENCY—Specifies the VCO frequency as a multiple of the input frequency. • MANUAL_PHASE—Specifies the VCO frequency as a phase shift step value.
VCO_MAX	Integer	Specifies the maximum value for the VCO pin. Use for simulation purposes only.

Parameter	Type	Description
VCO_MIN	Integer	Specifies the minimum value for the VCO pin. Use for simulation purposes only.
VCO_MULTIPLY_BY	Integer	Specifies the integer multiplication factor for the VCO pin. If omitted, the default is 0.
VCO_PHASE_SHIFT_STEP	Integer	Specifies the phase shift for the VCO pin. If omitted, the default is 0.
VCO_POST_SCALE	Integer	Specifies the VCO operating range. The VCO post-scale divider value is 1 or 2. If omitted, the default is 1.
WIDTH_CLOCK	Integer	Specifies the clock width. Values are 10 for Stratix III devices, 5 for Cyclone III and Cyclone IV devices, and 6 for all other supported device families. If omitted, the default is 6. For Stratix III, Stratix IV, Cyclone III, and Cyclone IV devices, the WIDTH_CLOCK parameter is required for both clear box and non-clear box implementation to reflect the correct width.

Related Information

- **ALTPLL Input Ports** on page 38
- **ALTPLL Output Ports** on page 41
- **ALTPLL Bidirectional Port** on page 46

Design Examples

The design examples use the ALTPLL IP core to:

- Generate an external differential clock from an enhanced PLL.
- Generate and modify internal clock signals.

These examples use the IP Catlog and parameter editor. When you complete the examples, you can incorporate them into your projects.

Related Information

- **Design Example 1: Differential Clock** on page 62
- **Design Example 2: Generating Clock Signals** on page 64

Design Files

The design files are available on the Literature and Technical Documentation page of the Altera website.

Related Information

- **ddr_clk.zip**

- ddr-clk-msim.zip
- shift_clk.zip
- shift_clk_msim.zip

Design Example 1: Differential Clock

This design example uses the ALTPLL IP core to generate an external differential clock from an enhanced PLL. You must generate or modify clock signals to meet design specifications. When you interface to double data rate (DDR) memory, you must generate a differential SSTL clock signal for the external device. A DDR DIMM requires three pairs of differential SSTL clocks. You can use enhanced PLLs in Stratix devices to generate these clock signals.

In this example, perform the following activities:

- Generate a 166-MHz differential SSTL external clock (`ddr_clk`) output from a 33.33-MHz input clock using the ALTPLL IP core and the parameter editor.
- Implement the `DDR_CLK` design by assigning the EP1S10F780 device to the project and compiling the project.
- Simulate the `DDR_CLK` design.

Generating a 166-MHz Differential SSTL External Clock

To generate a 166-MHz Differential SSTL external clock, follow these steps:

Before you begin

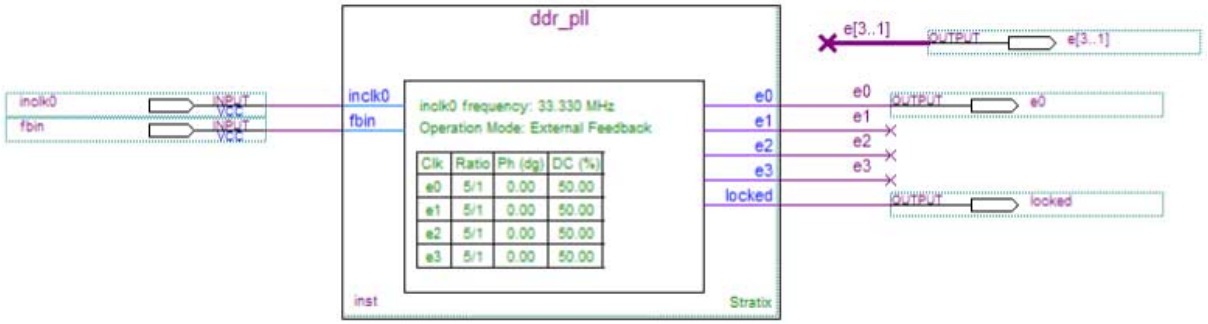
Download and unzip the ddr_clk.zip.

1. In the Quartus II software, open the project file **\ddr_clk\ddr_clk.qpf**.
2. Open the top-level **\ddr_clk\ddr_clk.bdf** file. You will complete this project in this example.
3. Click **Tools** > **IP Catalog**. Then expand the **I/O** folder and select **ALTPLL**
4. In **Which type of output file do you want to create?**, select **AHDL**.
5. For **What name do you want for the output file?**, name the output file **ddr_pll**.
6. For **What is the frequency of the inclock0 input?**, type `33.33`, and select **MHz**.
7. Under **PLL type**, click **Select the PLL type automatically**.
8. Under **Operation mode**, select **Create an 'fbin' input for an external feedback (External Feedback Mode)**.
9. Under **Operation mode**, for **Which output clock will have a board level connection?**, select **e0** from the drop-down menu.
10. In the **Dynamic configuration** section, turn off **Create optional inputs for dynamic reconfiguration**.
11. In the **Optional inputs** sections:
    a. Turn on **Create an 'pllena' input to selectively enable the PLL**.
    b. Turn on **Create an 'areset' input to asynchronously reset the PLL.**
    c. Turn off **Create an 'pfdena' input to selectively enable the phase/frequency detector**.
12. In the **Lock output** section, turn on **Create 'locked' output**.
13. Leave the remaining options at their default settings.
14. Click the **Output Clocks** tab.
15. Click **extclk e0**.

16. Turn on **Use this clock**.
17. Turn on **Enter output clock parameters**, and in the **Clock multiplication factor** box, type `5`.
18. In the **Clock division factor** box, type `1`.
19. In the **Clock duty cycle (%)** box, type `50.00`.
20. Click **Next**.
21. On page 14, repeat **step 16** through **step 19** for **extclk e1**.
22. Click **Next**.
23. On page 15, repeat **step 16** through **step 19** for **extclk e2**.
24. Click **Next**.
25. On page 16, repeat **step 16** through **step 19** for **extclk e3**.
26. Click **Next**. Page 17 appears. No input is required for this page.
27. Click **Next**. Page 18 appears.
28. On page 18, ensure that the Text Design File (**.tdf**), Pin Planner File (**.ppf**), AHDL Include File (**.inc**), Block Symbol File (**.bsf**), and Sample waveforms in summary file (**.html** and **.jpg**) are turned on.
29. Click **Finish**. The **ddr_pll** module is built.
30. In the **Symbol** dialog box of the **.bdf** file, click **OK**.
31. Move the pointer to place the **ddr_pll** symbol between the input and output ports in the **ddr_clk.bdf** file, connecting the inputs and outputs to the symbol. Click to place the symbol.
32. Save the design.

The following figure shows a completed design file.

Figure 28: ALTPLL ddr_pll Design Schematic



Implementing the ddr_clk Design

To assign the EP1S10F780 device to the project and compile the project, follow these steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, click **Device**. Select **Stratix** in the Device Family field.
3. In the **Target device** section, under **Available devices**, select **EP1S10F780C5**.
4. Click **OK**.
5. On the Processing menu, click **Start Compilation**.
6. When the **Full Compilation was successful** message box appears, click **OK**.
7. To view how the module is implemented in the Stratix device, on the Assignments menu, click **Timing Closure Floorplan**.

The `ddr_clk` design is now implemented.

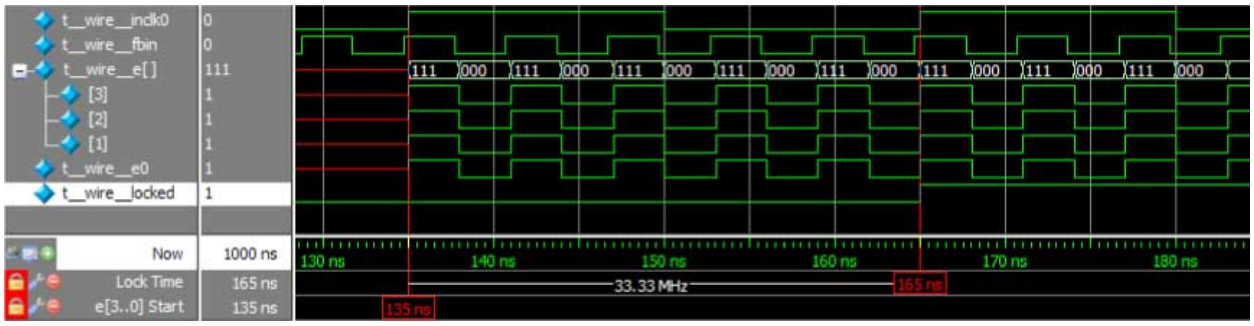
Functional Results — Simulate the ddr_clk Design in the ModelSim®-Altera Software

This ModelSim design example is for the ModelSim-Altera (Verilog) version. To simulate the design in the ModelSim-Altera software, follow these steps:

1. Unzip **ddr-clk-msim.zip** file to any working directory on your PC.
2. Locate the folder in which you unzipped the files to and open the **DDR_CLK.do** file in a text editor.
3. In line 1, replace <*insert_directory_path_here*> with the directory path of the appropriate library files. For example, `C:/Modeltech_ae/altera/verilog/stratix`
4. On the File menu, click **Save**.
5. Start the ModelSim-Altera software.
6. On the File menu, click **Change Directory**.
7. Select the folder in which you unzipped the files. Click **OK**.
8. On the Tools menu, click **TCL**, and click **Execute Macro**.
9. Select the **DDR_CLK.do** file and click **Open**. The **DDR_CLK.do** file is a script file for the ModelSim-Altera software to automate all the necessary settings for the simulation.
10. Verify the results shown in the Waveform Viewer window.

The following figure shows the expected simulation results in the ModelSim-Altera software.

Figure 29: ModelSim Simulation Results



Design Example 2: Generating Clock Signals

This design example uses the ALTPLL IP core to generate and modify internal clock signals. This example generates three internal clock signals from an external 100-MHz clock signal.

In this example, you perform the following activities:

- Generate 133 MHz, 200 MHz, and 200 MHz clocks that are time shifted by 1.00 ns from a 100 MHz external input clock using the ALTPLL IP core.
- Implement the `shift_clk` design by assigning the EP1S10F780 device to the project and compiling the project.
- Simulate the `shift_clk` design.

Generating 133-MHz, 200-MHz, and 200-MHz Time-Shifted Clocks

To generate 133-MHz, 200-MHz, and 200-MHz time-shifted clocks, follow these steps:

Before you begin

Download and unzip the **shift_clk.zip**.

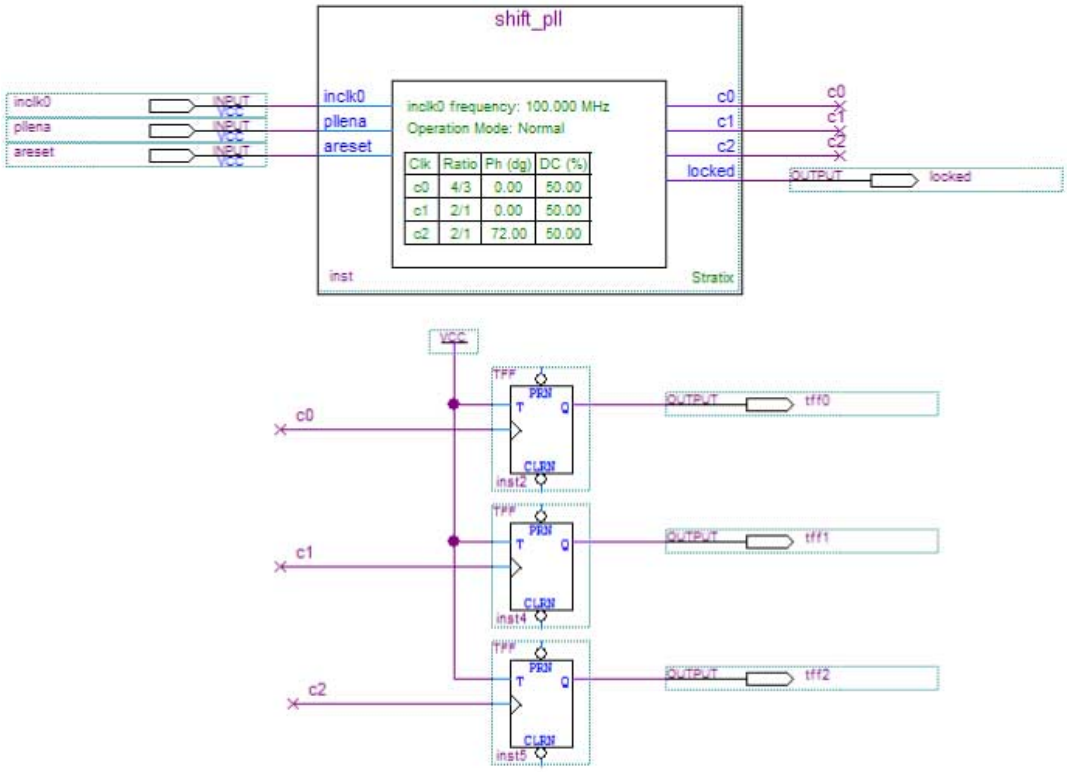
1. In the Quartus II software, open the project file **shift_clk.qpf**.
2. Open the top-level **shift_clk.qpf** file. You complete this project in this example.
3. Expand the **I/O** folder and click **ALTPLL**.
4. In **Which type of output file do you want to create?**, select **AHDL**.
5. For **What name do you want for the output file?**, name the output file as **shift_pll**.
6. Click **Next**. Page 3 appears.
7. To specify the 100 MHz input clock (`inclk0`), perform the following steps:
8. On page 3 of the **General** section, for **What is the frequency of the inclock0 input?,** type `100` **,** and select **MHz**.
9. Under **PLL type**, click **Select the PLL type automatically**.
10. In the **Operation mode** section, select **In Normal Mode**.
11. For **Which output clock will be compensated for?**, select **c0**.
12. Click **Next**. Page 4 appears.
13. In the **Dynamic configuration** section, turn off the **Create optional inputs for dynamic reconfiguration** option.
14. In the **Optional inputs** section, perform the following steps:
    a. Turn off **Create an 'pllena' input to selectively enable the PLL**.
    b. Turn off **Create an 'areset' input to asynchronously reset the PLL**.
    c. Turn off **Create an 'pfdena' input to selectively enable the phase/frequency detector**.
15. In the **Lock output** section, turn on **Create 'locked' output**.
16. Leave the other options in the **Parameter Settings** tab as the default.
17. To specify the 133 MHz output clock (`c0`), perform the following steps:
18. Click the **Output Clocks** tab to see the output clocks of the PLL. Page 7 appears.
19. On the **clk c0** page, turn on **Use this clock**.
20. In the **Clock Tap Settings** section, perform the following steps:
    a. Turn off **Enter output clock frequency**.
    b. Turn on **Enter output clock parameters**.
    c. For **Clock multiplication factor**, type `4`.
    d. For **Clock division factor**, type `3`.
    e. For **Clock phase shift**, type `0` and select **deg**.
    f. For **Clock duty cycle (%)**, type `50.00`.
21. Leave the other options at their default values.
22. Click **Next**. Page 8 appears.
23. To specify the 200 MHz output clock (`c1`), perform the following steps:
24. On the **clk c1** page, turn on **Use this clock**.
25. In the **Clock Tap Settings** section, perform the following steps:
    a. Turn off **Enter output clock frequency**.
    b. Turn on **Enter output clock parameters**.

c. For **Clock multiplication factor**, type `2`.
d. For **Clock division factor**, type `1`.
e. For **Clock phase shift**, type `0.00` and select **ns**.
f. For **Clock duty cycle (%)**, type `50.00`.

26. Leave all other options at their default values.
27. Click **Next**. Page 9 appears.
28. To specify the 200 MHz output clock `c2` with a 1.00 nanosecond delay, perform the following steps:
29. On the **clk c2** page, turn on **Use this clock**.
30. In the **Clock Tap Settings** section, perform the following steps:
    a. Turn off **Enter output clock frequency**.
    b. Turn on **Enter output clock parameters**.
    c. For **Clock multiplication factor**, type `2`.
    d. For **Clock division factor**, type `1`.
    e. For **Clock phase shift**, type `1.00` and select **deg**.
    f. For **Clock duty cycle (%)**, type `50.00`.
31. Leave all other options at their default values.
32. Click **Finish**. Page 18 appears.
33. On page 18, ensure that the Text Design File (**.tdf**), Pin Planner File (**.ppf**), AHDL Include file (**.inc**), Block Symbol File (**.bsf**), and Sample waveforms in summary file (**.html** and **.jpg**) are turned on.
34. Click **Finish**. The **shift_pll** module is built.
35. In the **Symbol** dialog box, click **OK**.
36. Move the pointer to place the **shift_pll** symbol between the input and output ports in the **shift_clk.bdf**. Click to place the symbol. You have now completed the design file.
37. On the File menu, click **Save Project** to save the design.

The following figure shows the completed design file.

Figure 30: ALTPLL shift_pll Design Schematic



Implementing the shift_clk Design

To assign the EP1S10F780C5 device to the project and compile the project, follow these steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the Category list, click **Device**. Select **Stratix** in the Device Family field.
3. In the Target device section, under the Available devices list, select **EP1S10F780C5**.
4. Click **OK**.
5. On the Processing menu, click **Start Simulation**.
6. When the **Full Compilation was successful** message box appears, click **OK**.
7. To view how the module is implemented in the Stratix device, from the Assignments menu, click **Timing Closure Floorplan**.

The `shift_clk` design is now implemented.

Simulating the shift_clk Design in the ModelSim-Altera Software

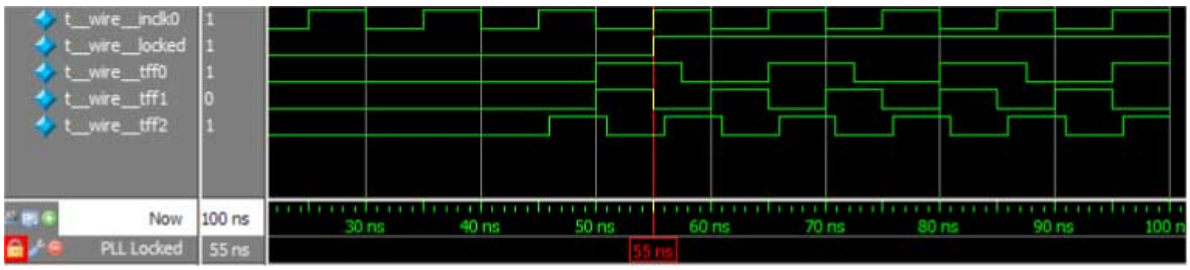
This ModelSim design example is for the ModelSim-Altera (Verilog) version. To simulate the design in the ModelSim-Altera software, follow these steps:

1. Download and unzip the **shift_clk_msim.zip** fileto any working directory on your PC.
2. Locate the folder in which you unzipped the files to, and open the **shift_clk.do** file in a text editor.
3. In line 1 of the **shift_clk.do** file, replace <*insert_directory_path_here*> with the directory path of the appropriate library files. For example, `C:/Modeltech_ae/altera/verilog/stratix`.

4. On the File menu, click **Save**.
5. Start the ModelSim-Altera software.
6. On the File menu, click **Change Directory**.
7. Select the folder in which you unzipped the files. Click **OK**.
8. On the Tools menu, click **TCL** and click **Execute Macro**.
9. Select the **shift_clk.do** file and click **Open**. The **shift_clk.do** fileis a script file for the ModelSim-Altera software to automate all the necessary settings for the simulation.
10. The Waveform Viewer appears. Verify the waveform shown in the viewer.

The following figure shows the expected simulation results in the ModelSim-Altera software.

Figure 31: ModelSim Simulation Results



Document Revision History

Lists the revision history for this user guide.

Table 20: Document Revision History

Date	Version	Changes Made
August, 2014	2014.08.18	• Updated parameterization steps for legacy parameter editor. • Added note that this IP core does not support Arria 10 designs.
June 2014	2014.06.30	• Replaced MegaWizard Plug-In Manager information with IP Catalog. • Added standard information about upgrading IP cores. • Added standard installation and licensing information. • Removed outdated device support level information. IP core device support is now available in IP Catalog and parameter editor. • Updated description in Operation Modes section. • Added Cyclone IV device in PLL Dynamic Reconfiguration Feature Support table. • Updated the Advanced Control Signals summary table.
August 2013	2013.08.22	Added a link to *AN454: Implementing PLL Reconfiguration in Stratix III and Stratix IV Devices*.
May 2013	9.0	DITA conversion.
November 2009	8.0	Updated to reflect new document organization and format.

Date	Version	Changes Made
December 2008	7.0	• Updated the following sections: • "Device Family Support" section • "Introduction" section • "Features" section • "General Description" section • "Design Examples" section • "Simulation" section • "Ports and Parameters" section • "How to Contact Altera" section • Removed the following sections: • "Resource Utilization & Performance" section • "Software and System Requirements" section • "Instantiating Multifunction in HDL Code" section • "Identifying a Multifunction after Compilation" section • "Signature II Embedded Logic Analyzer" section • Reorganized the "Using the MegaWizard Plug-In Manager" section into table format. • Renamed "About this User Guide" section to "Additional Information" and moved the section to the end of the user guide.
March 2007	6.0	Updated for software version 7.0, including: • Added support for Cyclone III devices • Added Referenced Documents section
December 2006	5.0	Updated to reflect new document organization, additions, and GUI changes for Quartus 6.1, including adding information relating to Stratix® III devices
May 2006	4.0	Updated to reflect new document organization, additions, and GUI changes for Quartus 6.0
December 2004	3.0	Updated to reflect new document organization and GUI changes