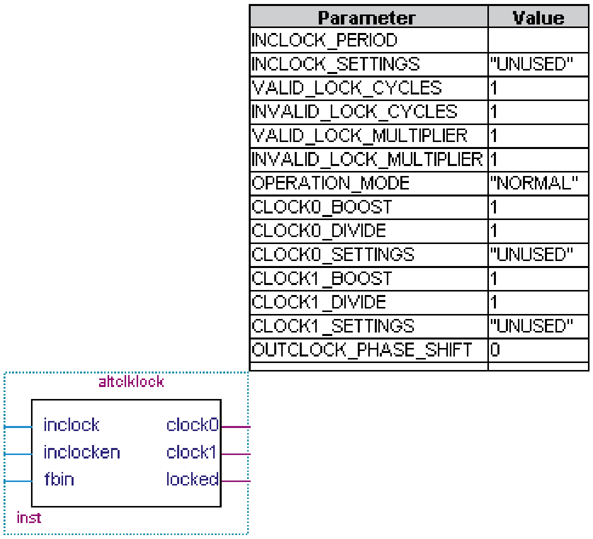
Using APEX 20K & APEX 20KE PLLs in the Quartus Software

Introduction

APEX™ devices support the ClockLock™, ClockBoost™, and ClockShift™ clock management features, which are implemented with phase-locked loops (PLLs). The ClockLock circuitry uses a synchronizing PLL that reduces the clock delay and skew within a device. This reduction minimizes clock-to-output and setup times while maintaining zero hold times. The ClockBoost circuitry, which provides a clock multiplier, allows designers to scale, or multiply, the input clock by integers or fractional ratios. The ClockShift circuitry provides programmable clock delay for phase shift applications or clock delay control for meeting strict timing requirements. The ClockLock, ClockBoost, and ClockShift features work in conjunction with the APEX device's high-speed clock to provide significant improvements in system performance and bandwidth.

You can use the `altclklock` megafunction within the Quartus™ software to enable the ClockLock, ClockBoost, and ClockShift features in APEX devices (see Figure 1). This document describes how to implement these clock management features using the `altclklock` megafunction in the Quartus software.

Figure 1. altclklock Megafunction

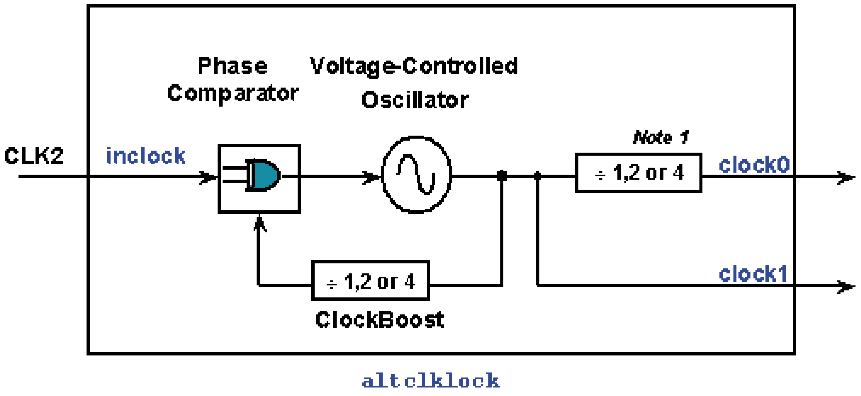


Parameter	Value
INCLOCK_PERIOD	
INCLOCK_SETTINGS	"UNUSED"
VALID_LOCK_CYCLES	1
INVALID_LOCK_CYCLES	1
VALID_LOCK_MULTIPLIER	1
INVALID_LOCK_MULTIPLIER	1
OPERATION_MODE	"NORMAL"
CLOCK0_BOOST	1
CLOCK0_DIVIDE	1
CLOCK0_SETTINGS	"UNUSED"
CLOCK1_BOOST	1
CLOCK1_DIVIDE	1
CLOCK1_SETTINGS	"UNUSED"
OUTCLOCK_PHASE_SHIFT	0

APEX 20K Devices

APEX 20K devices have one PLL that features ClockLock and ClockBoost circuitry. This PLL can be instantiated by using the `altclklock` megafunction. APEX 20K devices support ClockBoost multiplication circuitry, offering 1×, 2×, and 4× clock multiplication. Figure 2 shows the ClockLock and ClockBoost circuitry block diagrams within the `altclklock` megafunction and its ports.

Figure 2. altclklock Port-to-PLL Relationship for APEX 20K Devices



Note 1 - This division is used only for the purpose of dividing down a x2 / x4 clock1 to obtain a x1/x2 / x4 on clock0

You can use a single output clock of 1×, 2×, 4×, or any combination of output clocks. All `altclklock` clock outputs are 50/50 duty cycle. Table 1 describes the clock multiplication combinations that the `altclklock` megafunction supports for APEX 20K devices.

Table 1. Multiplication Factor Combinations

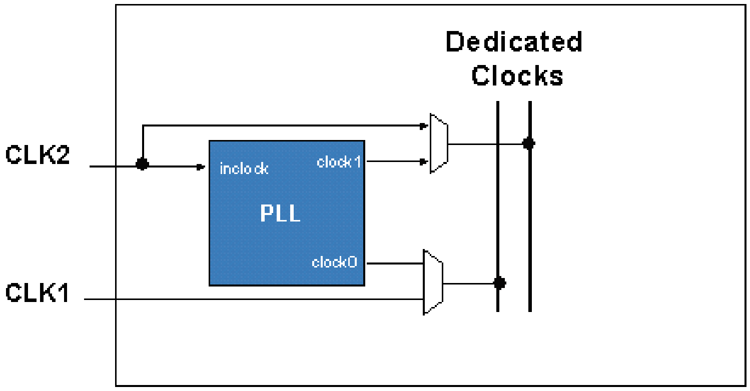
Clock0	Clock1	Input Frequency (MHz)
1×	1×	25 to 200
1×, 2×	2×	16 to 100
1×, 2×, 4×	4×	10 to 48

The dedicated clock pin (`CLK2`) supplies the clock to the PLL and the `altclklock` megafunction. The usage guidelines for the `altclklock` megafunction when used for APEX 20K devices are:

- The `inclock` port can only be fed directly by a dedicated clock input pin without inversion.
- The `altclklock` can only be used to clock positive or negative edge-triggered registers in LEs, IOEs, or ESBs.
- The `CLK2` pin that directly feeds the inclock port can also drive other registers without the PLL. However, doing so makes the `CLK1` pin and the `clock1` port unavailable.
- When two clock outputs are generated, the other clock pin (`CLK1`) cannot be used.

Connect the board clock trace to the `CLK2` pin only for designs that require two outputs from the `altclklock` megafunction. Figure 3 illustrates the valid clock connections for the PLL and the global clock lines.

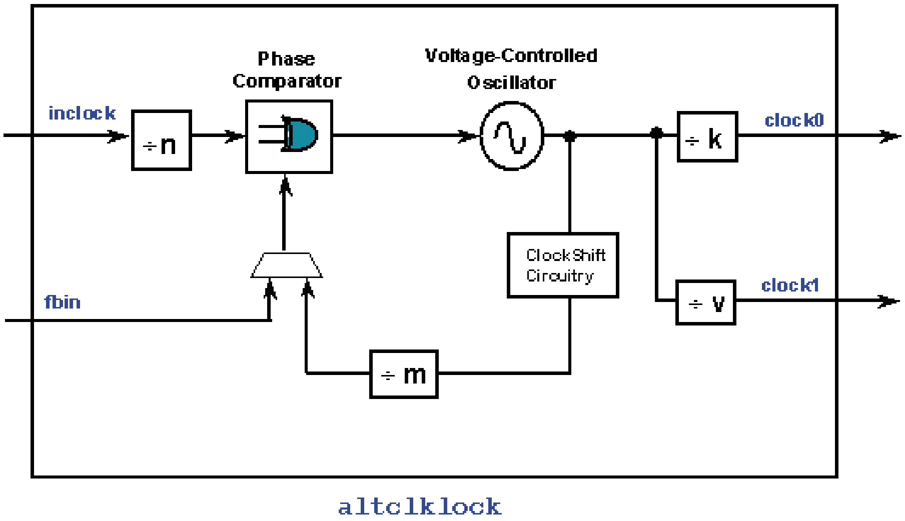
Figure 3. APEX 20K Dedicated Global Clock Pin Connections to PLL & Dedicated Clock Lines



APEX 20KE Devices

APEX 20KE devices incorporate multiple ClockLock circuits with advanced features. These features include ClockLock, advanced ClockBoost, low-voltage differential signaling (LVDS) support, ClockShift circuitry, and external clock outputs with optional external feedback inputs. Each APEX 20KE PLL includes circuitry that provides clock synthesis using $m/(n \times k)$ or $m/(n \times v)$ scaling. These scaling factors are chosen by the Quartus software according to the multiplication and division scaling parameter values in the `altclklock` instantiation. Figure 4 shows the ClockLock and ClockBoost circuitry in APEX 20KE devices.

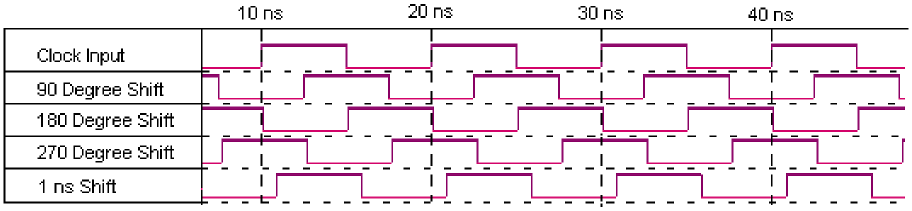
Figure 4. ClockLock & ClockBoost Circuitry in APEX 20KE Devices



Each of the dedicated global clock pins in EP20K300E, EP20K400E, EP20K600E, EP20K1000E, and EP20K1500E devices (`CLK1p`, `CLK2p`, `CLK3p`, `CLK4p`) supplies the clock to a PLL. Each `altclklock` instance represents a single PLL instantiation. All `altclklock` clock outputs are 50/50 duty cycle. The other general APEX 20KE usage guidelines for `altclklock` are:

- It can only be fed directly by a dedicated clock input pin without inversion.
- It can only be used to clock positive or negative edge-triggered registers in logic elements (LEs), input/output elements (IOEs), or embedded system blocks (ESBs).
- The allowable frequency input range is 1.5 to 160 MHz.
- The allowable frequency output range on `clock0` is 1.5 to 200 MHz.
- The allowable frequency output range on `clock1` is 20 to 200 MHz.
- Phase shifting is only possible on a multiplied clock if the input and output frequency have an integer multiple relationship, i.e. f_{in}/f_{out} or f_{out}/f_{in} must be an integer.
- Phase shifting, using degree or time units, will delay, or lag, the output clock with respect to the input clock (see Figure 5).
- The ratio of `clock_boost` to `clock_divide` cannot be greater than 160. There is also a special scaling ratio of 256/193 or 193/256 that is respectively allowed for T1/E1 or E1/T1 clock rate conversion.

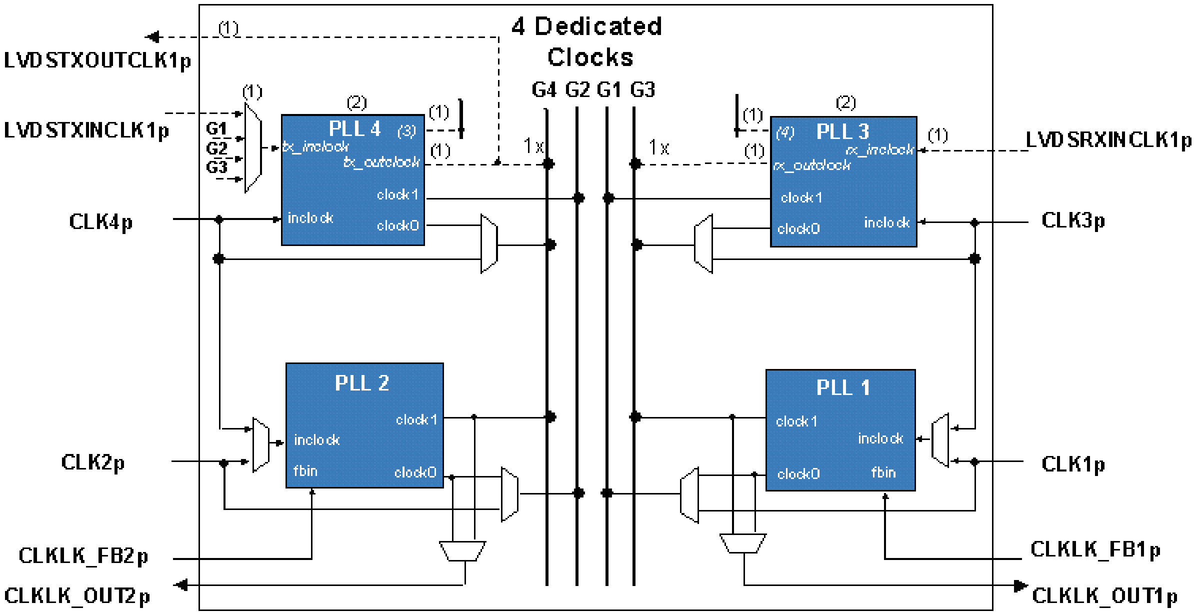
Figure 5. Phase & Delay Shifting Using APEX 20KE PLLs



There are several conditional equalities between input frequency, *m*, *n*, *k*, *v*, and phase shift values. The `altclklock` MegaWizard™ automatically sets the *m*, *n*, *k*, and *v* dividers to satisfy these equalities and to accommodate the clock multiplication or division and phase shift entered. The MegaWizard should be used to verify the validity of settings; the MegaWizard reports if a multiplication/division frequency ratio is not possible.

Each PLL can be driven by a dedicated clock pin and also bypassed simultaneously. `CLK3p` and `CLK4p` pins can feed two PLLs each, or two `altclklock` instances. This is useful for applications that need the phase shifted and non-phase shifted versions of the clock. Because the eight PLL outputs are shared among four possible dedicated global clock lines, certain combinations of multiple `altclklock` instances and their output connections are not possible. The Quartus software associates PLL numbers (1, 2, 3, 4) based on pin assignments made to the dedicated global clock pin that feed the `altclklock` megafunction. Figure 6 illustrates the valid clock connections for the PLL and the dedicated global clock lines in these devices. This figure should be used to determine whether a design clocking scheme is valid in terms of APEX 20KE PLL and clock connections. For example, `CLK4p` can feed PLL4 and PLL2 simultaneously, but only a single output from each PLL can be used since the four possible outputs feed two global clock lines.

Figure 6. Dedicated Global Clock Pin Connections to PLL & Dedicated Clock Lines for EP20K300E, EP20K400E, EP20K600E, EP20K1000E & EP20K1500E Devices



Notes:

(1) These PLL connections are used only in LVDS mode and apply to the `ALTLVDS` megafunction. `LVDSTXINCLK`, `LBDSRXINCLK`, and `LVDSTXOUTCLK` are designated dual purpose I/O pins that are used as LVDS clocks for the PLLs in LVDS mode.

(2) PLL3 or PLL4 can be configured for general purpose or LVDS use.

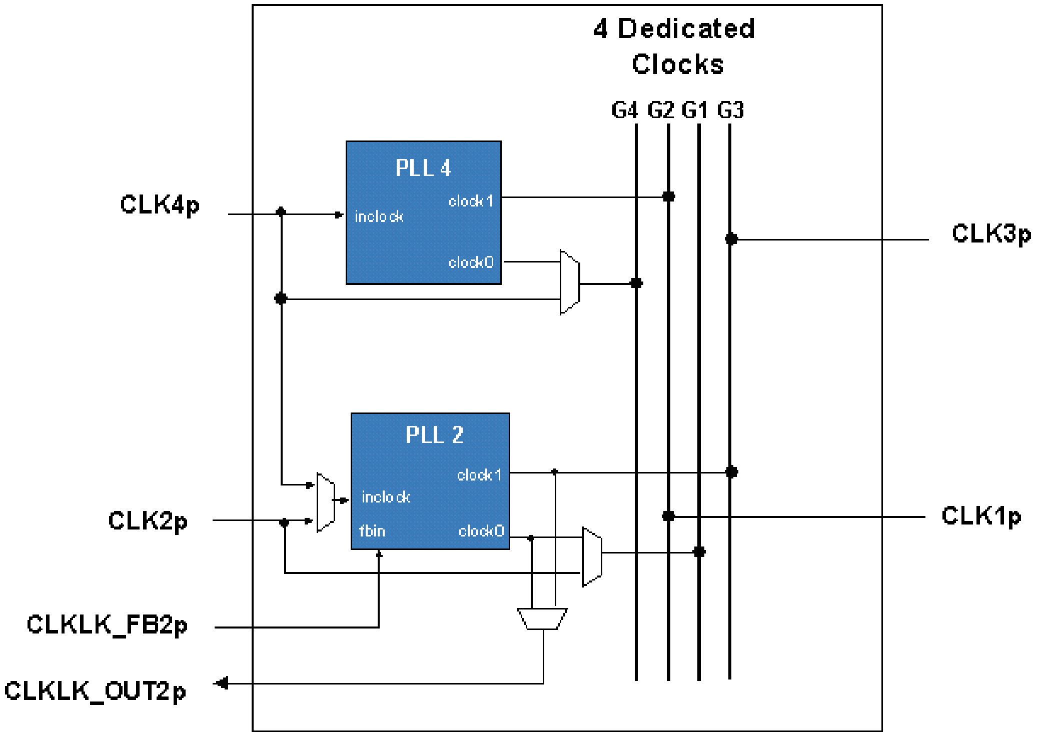
(3) This PLL is a high-speed CMOS/LVDS interface clock that feeds the LVDS transmitter block.

(4) This PLL is a high-speed LVDS/CMOS interface that feeds the LVDS receiver block.

For EP20K60E, EP20K100E, EP20K160E, and EP20K200E devices, the `CLK4p` and `CLK2p` dedicated clock pins supply the clock to two possible PLLs. These PLLs have the same usage guidelines as the EP20K400E and larger devices' PLLs, with the exception of some PLL connections. You can use all four possible PLL output clocks that are shared among four dedicated clock lines. If you use all the outputs, the `CLK3p` and `CLK1p` pins cannot be used.

Figure 7 illustrates the valid clock connections for the PLL and the global clock lines in these devices. Use Figure 7 to determine whether a design clocking scheme is valid in terms of PLL and clock connections. `altclklock` megafunction port connections should follow the usage guidelines illustrated in Figures 6 and 7.

Figure 7. Dedicated Global Clock Pin Connections to PLL & Dedicated Clock Lines for EP20K60E, EP20K100E, EP20K160E & EP20K200E Devices



altclklock Megafunction

The ClockShift and ClockBoost features, as well as other PLL feature settings, are controlled by the `altclklock` parameters. This section describes the ports and parameters for the `altclklock` megafunction and shows the function prototype and component declarations.

The following sample script shows an AHDL Function Prototype (port name and order also apply to Verilog HDL).

```
FUNCTION altclklock (inclock, inclocken, fbin)
   WITH (INCLOCK_PERIOD, INCLOCK_SETTINGS,
      VALID_LOCK_CYCLES, INVALID_LOCK_CYCLES, VALID_LOCK_MULTIPLIER,
      INVALID_LOCK_MULTIPLIER, OPERATION_MODE,
      CLOCK0_BOOST, CLOCK0_DIVIDE, CLOCK0_SETTINGS,
      CLOCK1_BOOST, CLOCK1_DIVIDE, CLOCK1_SETTINGS,
      OUTCLOCK_PHASE_SHIFT)
   RETURNS (clock0, clock1, locked);
```

```
VHDL Component Declaration:

COMPONENT altclklock
   GENERIC (INCLOCK_PERIOD: NATURAL;
      INCLOCK_SETTINGS: STRING := "UNUSED";
      VALID_LOCK_CYCLES: NATURAL := 3;
      INVALID_LOCK_CYCLES: NATURAL := 3;
      VALID_LOCK_MULTIPLIER: NATURAL := 1;
      INVALID_LOCK_MULTIPLIER: NATURAL := 1;
      OPERATION_MODE: STRING := "NORMAL";
      CLOCK0_BOOST: NATURAL := 1;
      CLOCK0_DIVIDE: NATURAL := 1;
      CLOCK1_BOOST: NATURAL := 1;
      CLOCK1_DIVIDE: NATURAL := 1;
      CLOCK0_SETTINGS: STRING := "UNUSED";
      CLOCK1_SETTINGS: STRING := "UNUSED";
      OUTCLOCK_PHASE_SHIFT: NATURAL := 0);

   PORT (inclock, inclocken: IN STD_LOGIC;
         fbin : IN STD_LOGIC := '0';
         clock0, clock1, locked : OUT STD_LOGIC);

END COMPONENT;
```

Tables 2 through 4 list `altclklock` input port, output port, and parameter descriptions.

Table 2. altclklock Input Port Descriptions

Port Name	Required	Description	Comments
`inclock`	Yes	Clock port that drives ClockLock PLL	
`inclocken`	No	PLL enable signal	When the `inclocken` port is high, the PLL drives the `clock0` and `clock1` ports. When the `inclocken` port is low, the `clock0` and `clock1` ports are driven by GND and the PLL goes out of lock. When the `inclocken` port goes high again, the PLL must relock. This port must be unconnected for APEX 20K devices.
`fbin`	No	External feedback input for PLL	To complete the feedback loop, there must be a board-level connection between the `fbin` pin and the external clock output pin of the PLL. Phase shifting on either clock output is not possible when external feedback is used. Division is still possible on both clock outputs, but multiplication is only possible on the clock output, `clock0`/`clock1`, which must be unconnected for APEX 20K devices.

Table 3. altclklock Output Port Descriptions

Port Name	Required	Description	Comments
`clock0`	No	First output clock of the PLL	In APEX 20K devices, if the pin driving the PLL `inclock` port is used somewhere else in the design, you can use only the PLL `clock0` output port. A deterministic no-fit occurs if you simultaneously use the `clock0` port, `clock1` port, and the pin driving the inclock port of the PLL. In APEX 20KE devices, you can use the `clock0` port, `clock1` port, and the pin driving the `inclock` port of the PLL in order to improve the fitting of the PLL. However, if you use the PLL to generate only one clock signal, use the `clock1` port to give the Compiler added flexibility when fitting the PLL.
`clock1`	No	Second output clock of the PLL	In APEX 20K devices, if the pin driving the `inclock` port of the PLL is used elsewhere in the design you can only use the `clock0` output port of the PLL. A deterministic no-fit occurs if you simultaneously use the `clock0` port, `clock1` port, and the pin driving the `inclock` port of the PLL. In APEX 20KE devices, you can use the `clock0` port, `clock1` port, and the pin driving the `inclock` port of the PLL in order to improve the fitting of the PLL. However, if you use the PLL to generate only one clock signal, use the `clock1` port to give the Compiler added flexibility when fitting the PLL.
`locked`	No	Status of PLL	When the PLL is locked, this signal is `VCC`; when the PLL is out of lock, this signal is `GND`. The locked port may pulse high and low while the PLL is in the process of achieving lock.

Table 4. altclklock Parameter Descriptions (Part 1 of 2)

Port Name	Type	Required	Description
INCLOCK_PERIOD	Integer	Yes	Specifies the period of the inclock port in ps or the frequency in MHz. MHz must be specified since ps is the default. This parameter is not required if a clock setting is specified for the inclock port.
INCLOCK_SETTINGS	String	No	Specifies the clock setting assignment to be used with the inclock port. If the INCLOCK_SETTINGS parameter is specified, the INCLOCK_PERIOD parameter is not required and is ignored. If the INCLOCK_SETTINGS parameter is omitted, the default is unused.
VALID_LOCK_CYCLES	Integer	No	Specifies the number of half-clock cycles that the clock0 and clock1 ports must be locked before the locked pin goes high. This parameter is only used for third-party and functional simulation. Use the output file from the MegaWizard Plug-In Manager to calculate the value for this parameter. If you do not enter a value for VALID_LOCK_CYCLES, the default is 5. This parameter is available only in APEX 20KE devices.
INVALID_LOCK_CYCLES	Integer	No	Specifies the number of half clock cycles that the clock0 and clock1 ports must be out of lock before the locked pin goes low. This parameter is used only for third-party and functional simulation. Use the output file from the MegaWizard Plug-In Manager to calculate the value for this parameter. If you do not enter a value for INVALID_LOCK_CYCLES, the default is 5. This parameter is available only in APEX 20KE devices.
VALID_LOCK_MULTIPLIER	Integer	No	Specifies the multiplier used along with internal PLL configuration information to generate the VALID_LOCK_CYCLES parameter. The actual VALID_LOCK_CYCLES value is displayed in the Compiler informational messages and in the MegaWizard output file. This parameter is required if the locked port is connected, and its values range from 1 to 5. If you do not enter a value for VALID_LOCK_MULTIPLIER, the default is 5. This parameter is available only in APEX 20KE devices.
INVALID_LOCK_MULTIPLIER	Integer	No	Specifies the multiplier used along with internal PLL configuration information to generate the INVALID_LOCK_CYCLES parameter. The actual INVALID_LOCK_CYCLES value is displayed in the Compiler informational messages and in the MegaWizard output file. This parameter is required if the locked port is connected, and its values range from 1 to 5. If you do not enter a value for INVALID_LOCK_MULTIPLIER, the default is 5. This parameter is available only in APEX 20KE devices.
OPERATION_MODE	String	No	In normal mode, if the PLL only feeds the internal clock network, the phase alignment occurs between the network and the dedicated inclock pin. If the PLL only feeds an external CLKLK_OUT pin, the phase alignment occurs between the external CLKLK_OUT pin and the dedicated inclock pin. If the PLL feeds both the internal clock network and an external clock CLKLK_OUT pin, the PLL compensates for the output pin, and the phase alignment occurs between the external CLKLK_OUT pin and the dedicated inclock pin. As a result, a phase error is introduced on the internal clock network. In NO_COMPENSATION mode, the PLL does not compensate for an external CLKLK_OUT clock pin. Therefore, there is always a phase error on the external CLKLK_OUT pin, regardless of whether the PLL drives the internal clock network. Values for the OPERATION_MODE parameter are NORMAL and NO_COMPENSATION. If you do not enter a value, the default is NORMAL. This parameter is available only in APEX 20KE devices. Also, LVDS is an allowable operation mode but cannot be specified by this megafunction—it is set with the altlvds megafunction.
CLOCK0_BOOST	Integer	No	Specifies the integer-multiplication factor for the clock0 port with respect to the input clock frequency. You can only specify this parameter, which must be greater than 0, if the clock0 port is used. However, it is not required if a clock setting is specified for the clock0 port. The value for this parameter must be 1, 2, or 4 for APEX 20K devices; for APEX 20KE devices, use the MegaWizard Plug-In Manager to calculate the value. If you do not specify a value, the default is 1.

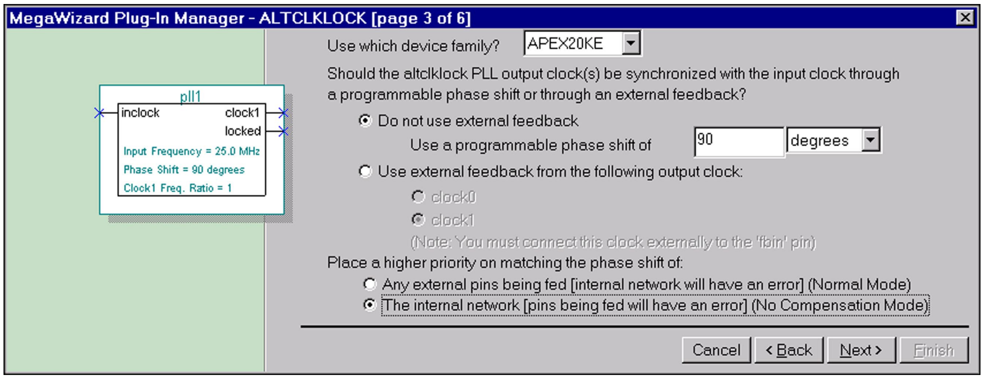
Table 4. altclklock Parameter Descriptions (Part 2 of 2)

Port Name	Type	Required	Description
CLOCK0DIVIDE	Integer	No	Specifies the integer division factor for the clock0 port with respect to the input clock frequency. You can specify this parameter, which must be greater than 0, only if the clock0 port is used. However, it is not required if a clock setting is specified for the clock0 port. The setting for this parameter must be 1 for APEX 20K devices. Use the MegaWizard Plug-In Manager to calculate the value for this parameter for APEX 20KE devices. If you do not specify a value, the default is 1.
CLOCK0_SETTINGS	String	No	Specifies the clock setting assignment to be used with the clock0 port. If this parameter is specified, the CLOCK0_BOOST, CLOCK0_DIVIDE, and OUTCLOCK_PHASE_SHIFT parameters are not required and are ignored. If both CLOCK0_SETTINGS and CLOCK1_SETTINGS are specified, they must have the same phase shift. If this parameter is not specified, the default is UNUSED.
CLOCK1_BOOST	Integer	No	Specifies the integer multiplication factor for the clock1 port with respect to the input clock frequency. You can only specify this parameter, which must be greater than 0, if the clock1 port is used. However, it is not required if a clock setting is specified for the clock1 port. The setting for this parameter must be 1, 2, or 4 for APEX 20K devices. Use the MegaWizard Plug-In Manager to calculate the value for this parameter for APEX 20KE devices. If this parameter is not specified, the default is 1.
CLOCK1_DIVIDE	Integer	No	Specifies the integer division factor for the clock1 port with respect to the input clock frequency. You can only specify this parameter, which must be greater than 0, if the clock1 port is used. However, it is not required if a clock setting is specified for the clock1 port. If this parameter is not specified, the default is 1.
CLOCK1_SETTINGS	String	No	Specifies the clock setting assignment used with the clock1 port. If this parameter is specified, the CLOCK1_BOOST, CLOCK1_DIVIDE, and OUTCLOCK_PHASE_SHIFT parameters are ignored. If both CLOCK0_SETTINGS and CLOCK1_SETTINGS are specified, they must have the same phase shift. If this parameter is not specified, the default is UNUSED.
OUTCLOCK_PHASE_SHIFT	Integer	No	Specifies the phase shift lag of the output clocks relative to the input clock, expressed as a time unit. Phase shifts of 0.0, 0.25, 0.5, or 0.75 times the input period (0, 90, or 270 degrees) are implemented precisely. The allowable range for the phase shift is between 0 ps and one input clock period. If the phase shift is outside this range, the Compiler adjusts it to fall within this range. For other phase shifts, the Compiler chooses the closest allowed value. If the fbin port is used, the programmable phase shift is not available. This parameter is not required if clock settings are used for the clock0 and clock1 ports. If this parameter is not specified, the default is 0. This parameter is only available in APEX 20KE devices.

MegaWizard Interface

The MegaWizard Plug-In Manager automatically sets the appropriate parameters. The options on page 3 of the MegaWizard window only apply to APEX 20KE PLLs, as shown in Figure 8. Table 5 lists the options available on page 3 of the altclklock MegaWizard Plug-In Manager.

Figure 8. Page 3 of the altclklock MegaWizard Plug-In Manager



MegaWizard Plug-In Manager - ALTCLKLOCK [page 3 of 6]

Use which device family? APEX20KE

Should the altclklock PLL output clock(s) be synchronized with the input clock through a programmable phase shift or through an external feedback?

● Do not use external feedback

Use a programmable phase shift of 90 degrees

○ Use external feedback from the following output clock:

○ clock0

● clock1

(Note: You must connect this clock externally to the 'fbin' pin)

Place a higher priority on matching the phase shift of:

○ Any external pins being fed [internal network will have an error] (Normal Mode)

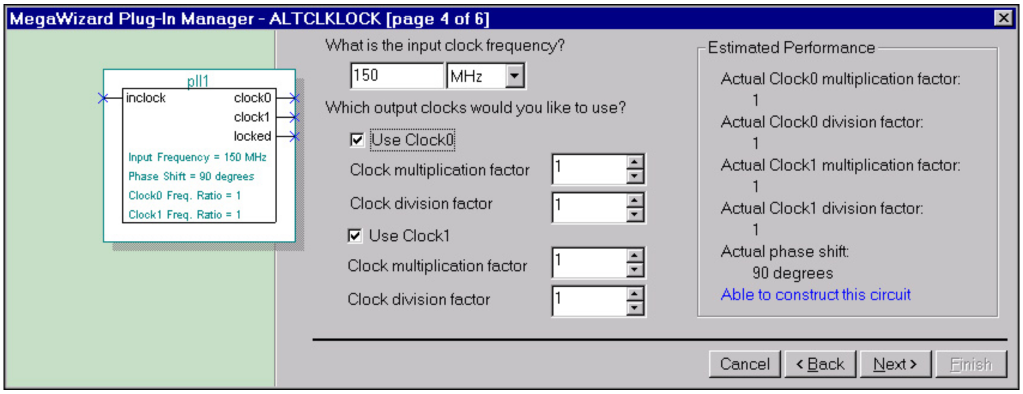
● The internal network [pins being fed will have an error] (No Compensation Mode)

Table 5. altclklock MegaWizard Plug-In Options

Option	Description
Do not use external feedback	If external feedback is not used, programmable phase shift is allowed. Any phase shift entered causes the output clock to lag the input clock. You can enter external feedback as degrees, ps, or ns by using the pull-down list. The smallest resolution that can be implemented is between 500 ps and 1 ns, depending on the voltage-controlled oscillator (VCO) frequency, which is not controlled by the user.
Use external feedback from the following output clock	If external feedback is used, programmable phase shift for clock outputs is disabled. Also, multiplication is disabled for the clock output that is chosen for external feedback. For example, if `clock0` is chosen as the external feedback output, then the `clock0` output cannot be multiplied, but `clock1` can be multiplied and used as an internal clock. Clock division is still possible for both clock outputs when external feedback is used.
Place a higher priority on matching the phase shift of: Any external pins being fed	This sets the Operation Mode parameter to `normal`. See Operation Mode description in Table 2.
Place a higher priority on matching the phase shift of: The internal network	This sets the Operation Mode parameter to `no_compensation`. The internal network is always aligned to the input; the external output pin does not have phase compensation, regardless of whether an internal clock is used. See Operation Mode description in Table 2.

Page 4 of the MegaWizard Plug-In Manager (see Figure 9) is for input frequency, clock multiplication, and clock division. The **Estimated Performance** box displays the actual multiplication, division, and phase shift. For circuits that can be constructed, the actual multiplication and division factors may differ from the values you enter, but the ratio of multiplication/division for a given clock output will be the same. For circuits that cannot be constructed, the closest achievable multiplication and division factors are displayed. The closest possible phase shift for the estimated performance ratios is also given. The inability to achieve the desired phase shift does not prevent circuit construction; the compiler achieves the closest possible shift, shown under **Actual phase shift** in the **Estimated Performance** box.

Figure 9. Page 4 of the altclklock MegaWizard Plug-In Manager



MegaWizard Plug-In Manager - ALTCLKLOCK [page 4 of 6]

What is the input clock frequency?

150 MHz

Which output clocks would you like to use?

☑ Use Clock0

Clock multiplication factor 1

Clock division factor 1

☑ Use Clock1

Clock multiplication factor 1

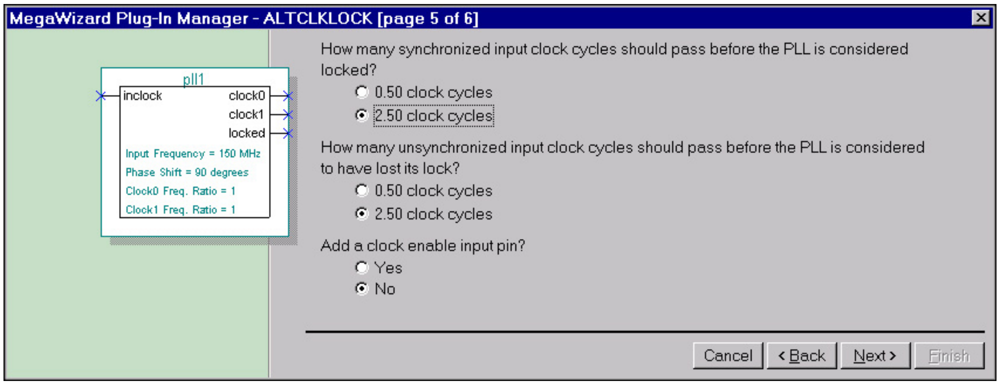
Clock division factor 1

Estimated Performance

Actual Clock0 multiplication factor:
1
Actual Clock0 division factor:
1
Actual Clock1 multiplication factor:
1
Actual Clock1 division factor:
1
Actual phase shift:
90 degrees
Able to construct this circuit

Page 5 of the MegaWizard Plug-In Manager (see Figure 10) provides options on user control for lock indication latency and the clock enable port. The radio button choices for the Lock indication are determined by internal PLL configuration parameters that are effected by the user-desired multiplication, division and frequency from the previous wizard pages. These options automatically set the `VALID_LOCK_CYCLES`, `INVALID_LOCK_CYCLE`, `VALID_LOCK_MULTIPLIER`, and `INVALID_LOCK_MULTIPLIER` parameters.

Figure 10. Page 5 of the altclklock MegaWizard Plug-In Manager



MegaWizard Plug-In Manager - ALTCLKLOCK [page 5 of 6]

How many synchronized input clock cycles should pass before the PLL is considered locked?

○ 0.50 clock cycles
● 2.50 clock cycles

How many unsynchronized input clock cycles should pass before the PLL is considered to have lost its lock?

○ 0.50 clock cycles
● 2.50 clock cycles

Add a clock enable input pin?

○ Yes
● No

Figures 11 and 12 are examples of PLL instantiations and configurations.

Figure 11. APEX 20K & APEX 20KE altclklock Instantiation with 2×, 4× Clocks & Clock Inversion

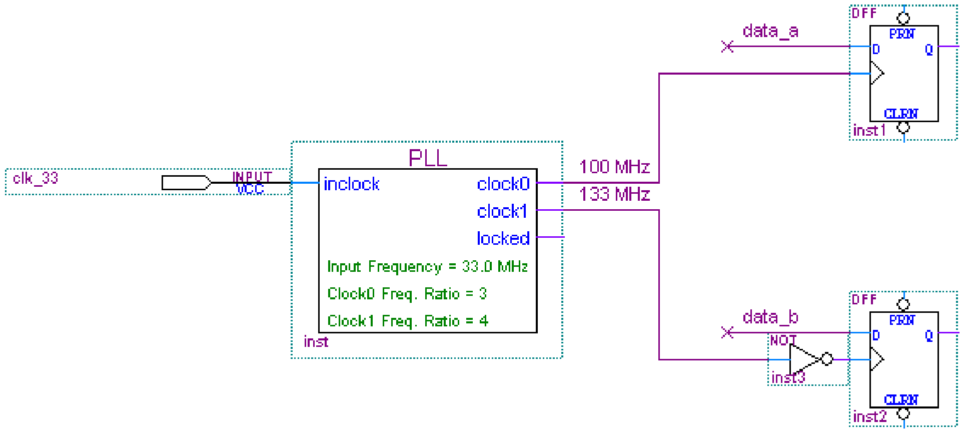
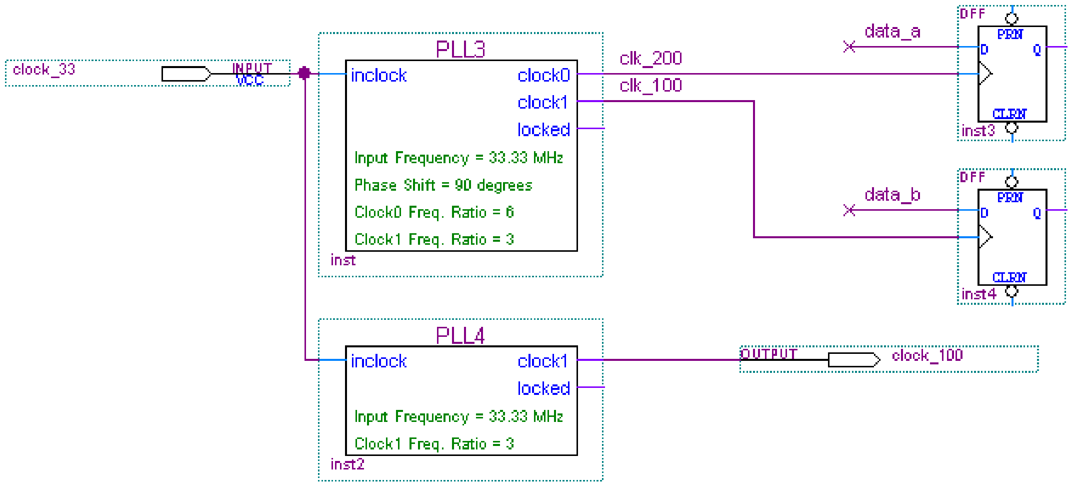


Figure 12. APEX 20KE altclklock Instantiation with Clock Multiplication, Phase Shift & External Clock Output



Reporting

The ClockLock section of the compilation report displays information regarding PLL usage (i.e., `altclklock` megafunction usage) in the device. This section is omitted if the design does not include PLLs. For more information on the ClockLock Section, see Quartus Help.

A compilation information message displays whether the requested `clock_boost` and `clock_divide` factors and/or the requested phase shift could be achieved. This information is useful if you do not use the MegaWizard Plug-In Manager to verify if a PLL configuration can be constructed. For unachievable `clock_boost` and `clock_divide` factors, compilation will fail with an error message displaying the closest achievable factors. For unachievable phase shift, the compilation displays the closest-achievable and implemented phase shift. Actual valid or invalid lock cycle indication is also displayed.

Timing Analysis

Multi-clock timing analysis causes the timing analyzer to report results using slack. The PLL input clocks and output clocks are different clocks that require multi-clock analysis. This is true even for the ×1 case because the clock coming out of the PLL is generated from the PLL VCO (not the clock pin), and a reduced-clock delay exists on the PLL output clock.

Another important fact is that the PLL is tuned to run at the frequency you specify or want. It will not function reliably when above or below the specified frequency (except for a 2.5% frequency tolerance). The PLL runs according to your specified settings and may not run at the maximum clock frequency (f_{MAX}). Because of this and the multi-clock analysis, f_{MAX} is not reported.

If f_{MAX} calculation is necessary, you can derive it from the reported slack. The micro t_{CO}, t_{SU}, and the path delay are given for a list path command on the Actual Maximum P2P timing in the Slack Report window. These can be added and inverted to find the f_{MAX} of that path.

When using an external feedback input, the External Input Delay option can be used to specify the amount of board delay from the external clock output pin back to the external feedback input. This assignment can be made on the pin through the Tools menu -> Assignment Organizer -> Timing.

Clock Domain Transfers

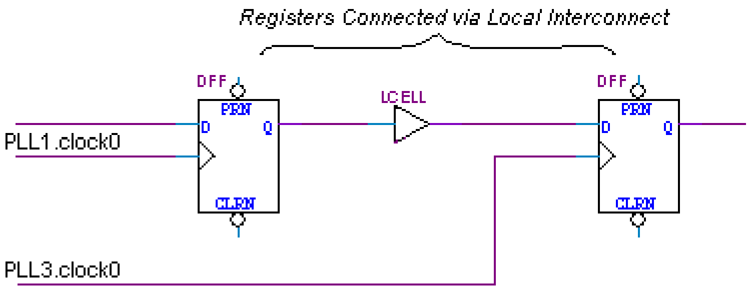
For data transfer across clock domains, specific design considerations should be made when using PLL clocks with synchronous and asynchronous transfers. The next two sections describe these considerations.

Synchronous Transfers

If the two clocks for domain transfer come from a single PLL, then all synchronous register-to-register transfers, (i.e. 50 MHz to 50 MHz or 50 MHz to 100 MHz), work across all conditions, and no special design considerations need to be made.

If the two clocks come from two different PLLs, (i.e., fed by the same clock with no ClockShift), then you must insert at least one logic element (LE) in the data path to guarantee data transfer between two registers that are connected via local interconnect. All other register-to-register transfers (e.g., across MegaLAB™ interconnects) work without special design considerations. Figure 13 shows the `LCELL` insertion for multiple clock source register-to-register transfer via local interconnect.

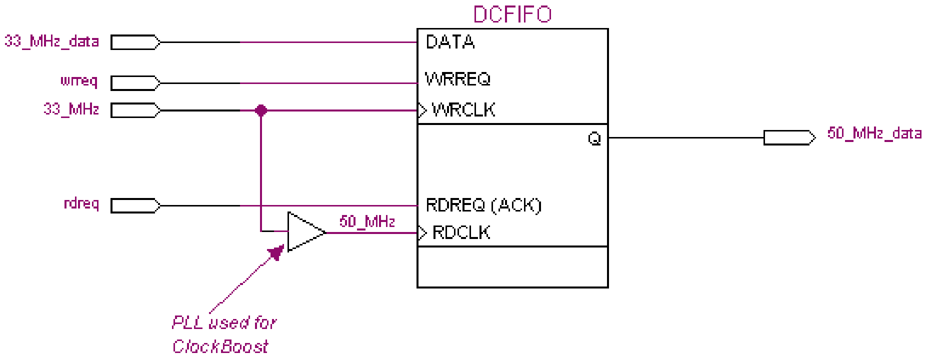
Figure 13. LCELL Insertion for Multiple PLL Clock Source Register-to-Register Transfer via Local Interconnect



Asynchronous Transfers

For asynchronous register-to-register transfer, (i.e., 50 MHz to 33 MHz), use the appropriate asynchronous design techniques to transfer data from one clock domain to the other. For example, the DCFIFO first-in first-out (FIFO) function can be used to buffer the data transfer. Figure 14 shows a DCFIFO that can be used to buffer the data transfer.

Figure 14. Using DCFIFO to Interface Between Asynchronous Clock Domains



If ClockShifted and non-ClockShifted clocks are used in a register-to-register transfer, the f_{MAX} may be reduced or a hold time violation may occur, depending on the direction and magnitude of the shift (any positive shift past 180 degrees can be considered negative shift) and whether the destination or source register's clock is shifted.

Simulation

The `altclklock` behavioral model can be used to simulate both the APEX20K PLL and the APEX20KE PLL by generating a clock signal based upon a reference clock. The APEX 20K and APEX 20KE behavioral models' instantiation should follow the same guidelines and restrictions as the design entry. The `altclklock` behavioral and timing models do not simulate jitter.

To simulate the External Feedback Input pin, the user provides the waveform expected for that pin. This waveform must have a frequency that is equal to the external output frequency (the input frequency divided by the `clock0/1` division factor, whichever is used for external feedback) with a delay that corresponds to the total delay from the output of the PLL to the External Feedback Input pin. This delay includes the PLL output to the External Clock Output pin delay (which can be obtained from Timing Analysis), as well as the anticipated board delay. This delay should not exceed 5 ns or 50% of the input clock period, whichever is less. When the External Feedback Input pin is being used, there is no compensation for the output pin delay.

The behavioral models for `altclklock` reside in the \quartus\eda\sim_lib directory. APEX20KE_MF.VHD contains the VHDL behavioral models and can be used for `altclklock` in both APEX 20K and APEX20KE devices. APEX20KE_MF.v contains the Verilog behavioral models and can be used for `altclklock` in both APEX 20K and APEX 20KE devices.

The behavioral model does not perform error checking, and the user must only specify valid values for the parameters of `altclklock`. When targeting APEX 20K devices, be sure to only use APEX 20K-applicable parameters with appropriate values.

In order to simulate the model successfully, the resolution of the VHDL simulator must be set to ps. A larger resolution will result in calculation rounding and thus create incorrect multiplication or division.

Sample VHDL Instantiation of the altclklock Model in a Design

The following shows a sample VHDL instantiation of the `altclklock` model in a design.

```
library ieee;
use ieee.std_logic_1164.all;

entity pll_design is
         port (inclock : in std_logic;
      inclocken : in std_logic;
                  data_in1 : in std_logic_vector(7 downto 0);
      clock0 : out std_logic;
      r_out: out std_logic_vector(7 downto 0);
      locked: out std_logic);
end pll_design;

architecture apex of pll_design is

component my_dff
         port (  clock  :  in STD_LOGIC;
                  data:  in STD_LOGIC_VECTOR(7 DOWNTO 0);
                  q  :  out STD_LOGIC_VECTOR(7 DOWNTO 0));
end component;

component altclklock
   generic (
         inclock_period : natural;
         inclock_settings : string := "UNUSED";
         valid_lock_cycles : natural := 5;
         invalid_lock_cycles : natural := 5;
         valid_lock_multiplier : natural := 5;
         invalid_lock_multiplier : natural := 5;
         operation_mode : string := "NORMAL";
         clock0_boost : natural := 1;
         clock0_divide : natural := 1;
         clock1_boost : natural := 1;
         clock1_divide : natural := 1;
         clock0_settings : string := "UNUSED";
         clock1_settings : string := "UNUSED";
         outclock_phase_shift : natural := 0 );

port (inclock : in std_logic;
      inclocken : in std_logic;
      fbin : in std_logic := '0';
      clock0 : out std_logic;
      clock1: out std_logic;
      locked: out std_logic);
end component;

signal clock1_sig:     std_logic;
begin
```

```
U0: altclklock
   generic map
(
      inclock_period => 40000,
      clock1_boost => 4,
      clock1_divide => 1,
      clock0_boost => 2,
      clock0_divide => 1,
      operation_mode => "NORMAL",
      valid_lock_cycles => 5,
      invalid_lock_cycles => 5,
      valid_lock_multiplier => 5,
      invalid_lock_multiplier => 5,
      outclock_phase_shift => 10000
)

   port map
      (inclock => inclock,
      inclocken => inclocken,
      clock0 => clock0,
      clock1 => clock1_sig,
      locked => locked);

process(clock1_sig)
begin
         if clock1_sig'event and clock1_sig = '1' then
            r_out <= data_in1;
         end if;

end process;

end apex;
```

Sample Testbench for the VHDL Design

The following shows a sample testbench for the VHDL design.

```
library ieee;
use ieee.std_logic_1164.all;

entity plltest2 is
end plltest2;

architecture behave2 of plltest2 is

      signal inclock : std_logic := '0';
      signal inclocken : std_logic;
      signal data_in1 : std_logic_vector(7 downto 0) := "10101010";
      signal clock0 : std_logic;
      signal locked : std_logic;
      signal r_out : std_logic_vector(7 downto 0);
      component pll_design
      port (
        inclock : in std_logic;
        inclocken : in std_logic;
        data_in1 : std_logic_vector(7 downto 0);
        clock0 : out std_logic;
        r_out : out std_logic_vector(7 downto 0);
        locked : out std_logic) ;
      end component;
begin

inclocken <= '1' after 5 ns;

U0 : pll_design port map (
        inclock => inclock,
        inclocken => inclocken,
        data_in1 => data_in1,
        clock0 => clock0,
        r_out => r_out,
        locked => locked);

process(inclock)
begin
   for i in 1 to 100 loop
      inclock <= not inclock after 20 ns;
   end loop;
end process;

end behave2;

configuration pllconfig of plltest2 is
   for  behave2
      for U0: pll_design use entity work.pll_design(apex);
      end for;
   end for;
end pllconfig;
```

Example Verilog Instantiation of the altclklock Model in a Design

The following shows an example Verilog instantiation of the `alkclklock` model in a design.

```
module pllsource (inclock, inclocken, data_in1, clock0, r_out, locked);
      input inclock, inclocken;
      input [7:0] data_in1;
      output clock0, locked;
      output [7:0] r_out;

wire clock1_sig;
reg [7:0] r_out;

altclklock PLL_1

      ( .inclock(inclock), .inclocken(inclocken), .clock0(clock0),
        .clock1(clock1_sig), .locked(locked));

defparam
            PLL_1.inclock_period = 50000,
            PLL_1.inclock_settings = "UNUSED",
            PLL_1.clock0_settings = "UNUSED",
            PLL_1.clock1_settings = "UNUSED",
            PLL_1.valid_lock_cycles = 5,
            PLL_1.invalid_lock_cycles = 5,
            PLL_1.valid_lock_multiplier = 5,
            PLL_1.invalid_lock_multiplier = 5,
            PLL_1.clock0_boost = 4,
            PLL_1.clock1_boost = 2,
            PLL_1.clock0_divide = 1,
            PLL_1.clock1_divide = 1,
            PLL_1.outclock_phase_shift = 0,
            PLL_1.operation_mode = "NORMAL";

always @(posedge clock1_sig)
begin
      r_out = data_in1;
end
```

Sample Testbench for Verilog Design

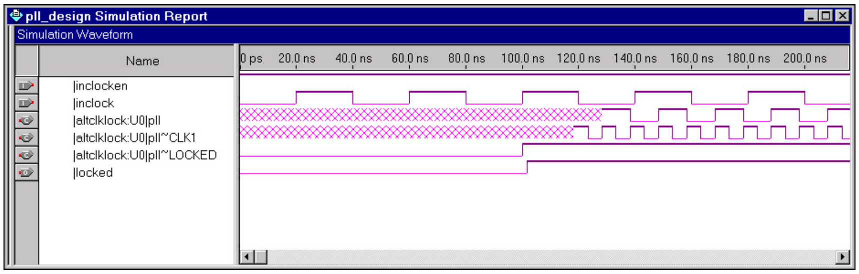
The following shows a sample testbench for Verilog design.

```
timescale 1 ns/100ps

module plltest;

parameter tmp = 8'b 10101010;
reg inclock, inclocken;
reg [7:0] data_in1;
wire clock0, locked;
wire [7:0] r_out;

pllsource U1

      ( .inclock(inclock), .inclocken(inclocken), .data_in1(data_in1),
        .clock0(clock0), .r_out(r_out), .locked(locked));

initial
      data_in1 = tmp;

initial
      inclock = 0;
always #25 inclock = ~inclock;

initial
      begin
         #0 inclocken = 0;
         #5 inclocken = 1;
      end

initial
      begin
         #100 data_in1 = 8'b 11110000;
         #200 data_in1 = 8'b 00110011;
      end

endmodule
```

Sample Waveform

Figure 15 shows an example waveform for dual clock outputs of the APEX 20KE PLL. In this example, `clock0` is a 2x clock and `clock1` is a 4x clock; both are shifted/lag by 90 degrees. For simulation, `|altclklock|`<*instance*>`|pll` is the `clock0` output of the PLL, `|altclklock|`<*instance*>`|pll~CLK1` is the `clock1` output of the PLL, and `|altclklock|`<*instance*>`|pll~LOCKED` is the locked output indication. In timing simulation, output clocks have a slight negative shift because they are at the output of the PLL and not at flip-flop clock ports. A positive delay is added as they reach clock ports of RAMs or flip-flops.

Figure 15. Timing Simulation Output Waveform for Dual-Output Clocks with 90° Shift



Summary

The advanced feature set of APEX 20K and APEX 20KE PLLs, such as ClockLock, ClockBoost and ClockShift, can be controlled with the `altclklock` megafunction in the Quartus software. The ClockLock, ClockBoost, and ClockShift features work in conjunction with APEX devices' high-speed clocks and provide significant improvements in system performance and bandwidth.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
http://www.altera.com

Copyright © 2000 Altera Corporation. Altera, APEX, ClockLock, ClockBoost, ClockShift, EP20K60E, EP20K100E, EP20K160E, EP20K200E, EP20K300E, EP20K400E, EP20K600E, EP20K1000E, EP20K1500E, MegaWizard, MegaLAB, and Quartus, are trademarks and/or service marks of Altera Corporation in the United States and other countries. Other brands or products are trademarks of their respective holders. The specifications contained herein are subject to change without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for