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. Use this application note in conjunction with the following literature:

- *Clock Networks and PLLs in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*

- *Clock Networks and PLLs in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*

- *Phase-Locked Loop (ALTPLL) Megafunction User Guide*

- *Phase-Locked Loop Reconfiguration (ALTPLL_RECONFIG) Megafunction* user guide

- *Internal Memory (RAM and ROM) User Guide*

- *External Memory PHY Interface (ALTMEMPHY) (nonAFI) Megafunction User Guide*

This application note discusses the following topics:

- Overview of PLL reconfiguration in Stratix III and Stratix IV devices

- Complete flow on implementing real-time PLL reconfiguration feature in a frequency pre-scaler application

- Complete flow on implementing dynamic phase shifting feature

- Design considerations for system designers when selecting PLL reconfiguration parameters

The design examples described in this application note are available for download from the Literature: Application Notes page on the Altera website. For instructions to download and use the design examples, refer to "Design Example 1" on page 11 and "Design Example 2" on page 15.

101 Innovation Drive
San Jose, CA 95134
www.altera.com

ISO
9001:2008
Registered

Feedback   Subscribe

Introduction

PLLs use several divide counters and different voltage controlled oscillator taps to perform frequency synthesis and phase shifts. In Stratix III and Stratix IV PLLs, you can reconfigure the counter settings and dynamically phase-shift the PLL output clock. You can also change the charge pump and loop filter components, which dynamically affect the PLL bandwidth. You can use these PLL components to update the clock frequency, PLL bandwidth, and phase shift in real time, without reconfiguring the entire FPGA.

Applications that operate at multiple frequencies can benefit from PLL reconfiguration in real time. PLL reconfiguration is also beneficial in prototyping environments, allowing you to sweep PLL output frequencies and adjusting the clock output phase at any stage of the design. For instance, a system generating test patterns is required to generate and transmit patterns at 50 or 100 MHz, depending on the device under test. Reconfiguring the PLL components in real time allows you to switch between two such output frequencies within a few microseconds. You can also use this feature to adjust clock-to-out (t_{CO}) delays in real time by changing output clock phase shift. This approach eliminates the need to regenerate a configuration file with new PLL settings.

PLL Reconfiguration in Stratix III and Stratix IV Devices

The following PLL components are reconfigurable in real time:

- Pre-scale counter (*n*)

- Feedback counter (*m*)

- Post-scale counters (C0-C9)

- Post VCO Divider (K)

- Charge-pump current (ICP), and loop-filter components (R, C)

Figure 1 shows how PLL counter settings can be dynamically adjusted by shifting their new settings into a serial shift-register chain or scan chain. Serial data is the input to the scan chain via `scandataport` and the shift registers are clocked by `scanclk`. Serial data is shifted through the scan chain as long as the `scanclkena` signal stays asserted. After the last bit of data is clocked, asserting the reconfiguration state machine signal, `configupdate`, for at least one `scanclk` cycle causes the PLL configuration bits to be synchronously updated with the data in the scan registers. The scan chain can also be initialized or changed using a Memory Initialization File (**.mif**) in Hexadecimal File (**.hex)** or **.mif** format. For more information about **.mif** settings, refer to "PLL Reconfiguration Scan Register Bitmap" on page 9.

Figure 1. PLL Reconfiguration Scan Chain [1], [3]
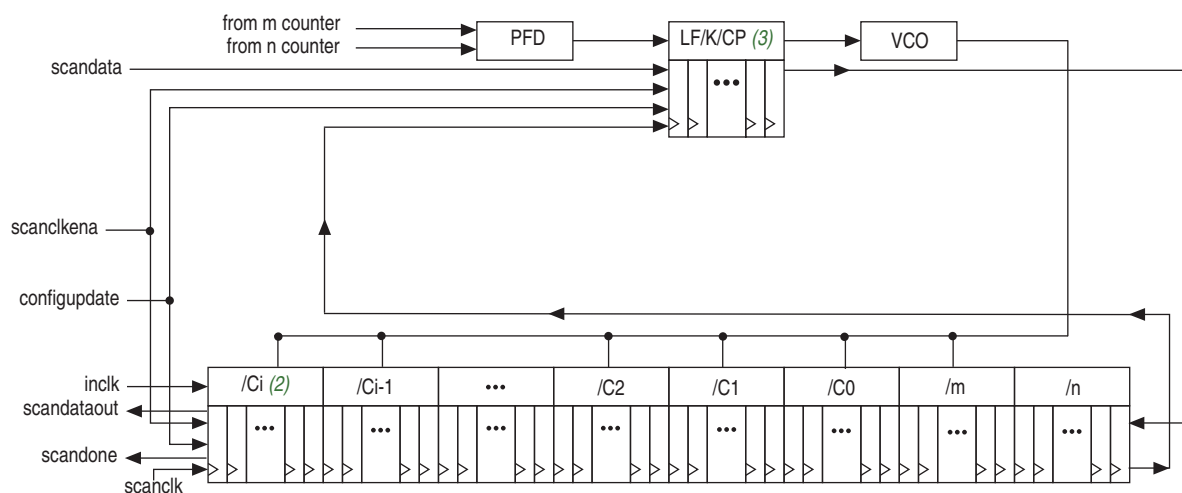
Notes to Figure 1:

(1) The Stratix III and Stratix IV Left/Right PLLs support C0 — C6 counters.

(2) i = 6 or i = 9.

(3) This figure shows the corresponding scan register for the K counter in between the scan registers for the charge pump and loop filter. The K counter is physically located after the VCO.

📖 For more information about the hardware and software implementation for Stratix III devices, refer to the *Clock Networks and PLLs in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook* and the *Phase-Locked Loop (ALTPLL) Megafunction User Guide*. For more information about the hardware and software implementation for Stratix IV devices, refer to the *Clock Networks and PLLs in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook* and the *Phase-Locked Loop (ALTPLL) Megafunction User Guide*.

Implementing PLL Reconfiguration using Quartus II Software

You can use the ALTPLL MegaWizard® Plug-In Manager to enable reconfiguration circuitry in the ALTPLL megafunction instantiation in your design. The ALTPLL_RECONFIG megafunction simplifies the process of reconfiguring Stratix III and Stratix IV PLLs.

📖 For more information, refer to the *Phase-Locked Loop Reconfiguration (ALTPLL_RECONFIG) Megafunction* user guide and the *Phase-Locked Loop (ALTPLL) Megafunction User Guide*, respectively.

There are two methods to reconfigure PLLs using the ALTPLL_RECONFIG megafunction.

■ Method 1—Reconfigure a specific PLL counter without affecting other PLL settings. Use this method when a few PLL settings must be modified at a time.

■ Method 2—Reconfigure all PLL counter settings using a **.mif**. The **.mif** updates the scan chain, which subsequently reconfigure the PLL counters.
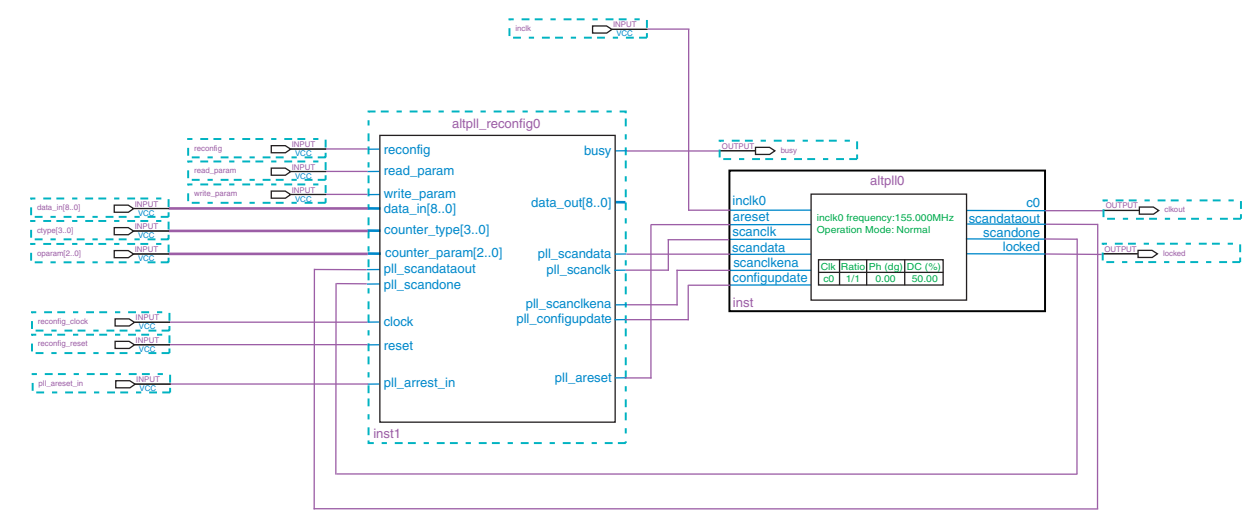
PLL Reconfiguration: Method 1

To reconfigure a specific PLL counter, shift in specific `counter_type`, `counter_param`, and `data_in` to the ALTPLL_RECONFIG megafunction. Figure 2 shows the connections between the ALTPLL megafunction and the ALTPLL_RECONFIG megafunction.

☞ In Quartus® II version 5.0 and later, the PRESERVE_PLL_COUNTER_ORDER setting in the Quartus II software logic option is automatically turned on if the PLL reconfiguration feature is used in the PLL. Because the software logic option is turned on, the Quartus II software does not rotate counters to improve clock routing, and some clock routing errors can occur as a result. To rectify this, you can first turn off the PLL reconfiguration and let the Quartus II software pick the optimal counter selection, then turn the PLL reconfiguration back on and change the PLL outputs to use the counter selection as chosen by the Quartus II software (by moving c0 to c1, etc.).

Figure 2. ALTPLL_RECONFIG and ALTPLL Megafunctions in the Quartus II Software

The following steps describe PLL reconfiguration using Method 1:

1. Choose the 4-bit `counter_type` (C0-C9, M, N, CP/LF, VCO).

   🔖 For valid `counter_type` settings, refer to the *Phase-Locked Loop Reconfiguration (ALTPLL_RECONFIG) Megafunction* user guide.

2. Choose the 3-bit `counter_param` for counter chosen in step 1. For valid parameter settings, refer to Table 1.

Table 1. Stratix III and Stratix IV ALTPLL_RECONFIG counter_param Settings

Counter Type	Counter_Param Name	Counter_Param Value	Data_in Width (bits)
C0-C9: Top/Bottom PLLs C0-C6: Left/Right PLLs	High count	000	8
	Low count	001	8
	Bypass	100	1
	Mode (odd/even division)	101	1
ChargePump/LoopFilter	Charge pump current	000	3
	Loop filter resistor	001	5
	Loop filter capacitor	010	2
VCO	VCO Post Scale	000	1
M/N counters	High count	000	8
	Low count	001	8
	Bypass	100	1
	Mode (odd/even division)	101	1
	Nominal count	111	9

3. Choose the 9-bit new value `data_in` for the `counter_param` in step 2.

4. Assert `write_param` for one `scanclk` cycle to allow input from steps 1 through 3 to be written to the scan chain.

5. The `busy` signal is asserted on the rising edge of `scanclk` following the assertion of `write_param`. The `busy` signal remains asserted while the parameter is being written. The time for which the `busy` signal is asserted varies depending on the data shifted into the scan chain.

6. When the `busy` signal is de-asserted, you can write a different `counter_type` into the scan chain following steps 1 through 5.

7. When you have shifted all the data into the scan chain, assert the `reconfig` signal after the `busy` signal has been de-asserted to reconfigure the PLL with the contents of the scan chain.

Figure 3 shows a timing diagram for reconfiguration using Method 1.

Figure 3. Timing Diagram for PLL Reconfiguration Using Method 1
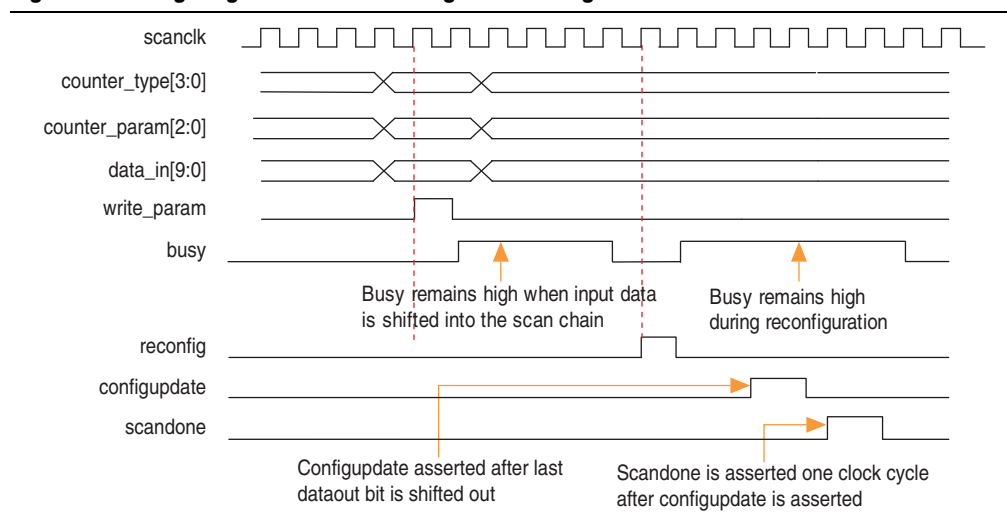
PLL Reconfiguration: Method 2

Initialize the ALTPLL_RECONFIG megafunction with a **.mif** in **.mif** or **.hex** format. The **.mif** file is a bitmap of the PLL reconfiguration scan chain. Pulse the reconfig signal to update the ALTPLL megafunction with the **.mif** contents.

The design setup is similar to Method 1, except that the counter_type, counter_param, and datain ports are not used in this example. It is possible to use Method 1 and Method 2 in the same design, provided the control signals are asserted properly.

To ensure that the **.mif** has valid settings, create a separate ALTPLL megafunction instantiation with the new input and output settings for your PLL after reconfiguration and select **Generate a .mif file** in the ALTPLL MegaWizard Plug-In Manager. Use the generated **.mif** for reconfiguration in your design.

For Stratix III and Stratix IV devices, multiple reconfigurations are possible using multiple **.mifs**. You can create multiple **.mifs** with the MegaWizard Plug-In Manager without having to recompile your design. For details, refer to "Example 1: Frequency Pre-scaler in a Multi-Rate Line Card" on page 11.

The following steps describe PLL reconfiguration using Method 2:

1. Create the **.mif** for the new configuration settings in **.mif** or **.hex** format.

2. Associate the ALTPLL_RECONFIG megafunction with the **.mif**.

3. Assert the reconfig signal to update the scan chain and the PLL counters with the contents of the **.mif**.

Figure 4 shows the timing diagram for reconfiguration using a **.mif** in a ALTPLL_RECONFIG megafunction.

Figure 4. Timing Diagram for Reconfiguration Using a .mif in the ALTPLL_RECONFIG Megafunction
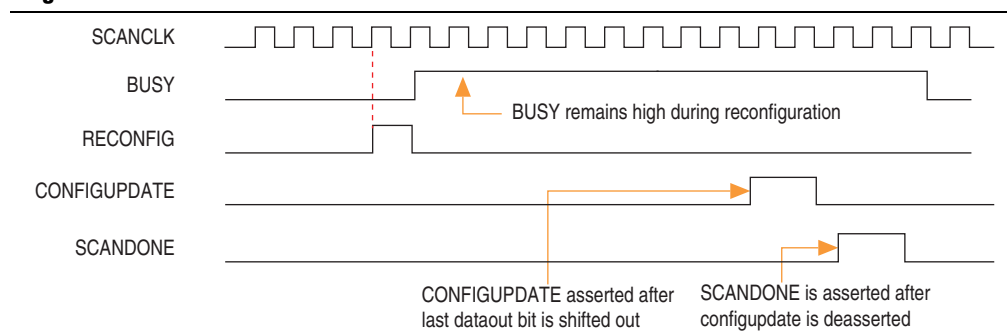
☞ You can open a **.mif** in a text editor to take advantage of the comments embedded in the file. These comments show you the scan chain values and positions based on your design parameterization. These comments are removed from the file if you open a **.mif** in the Quartus II software. If you have already viewed a **.mif** in the Quartus II software, you can regenerate it in the ALTPLL MegaWizard interface to restore the comments.

Reconfiguration Signals

The following list describes the reconfiguration signals:

■ The write_param signal is sampled at the rising edge of scanclk. The write_param signal needs to be asserted for only one scanclk cycle to prevent the parameter from being accidentally rewritten on any subsequent clock cycle.

■ The reconfig signal is sampled at the rising edge of scanclk. reconfig needs to be asserted only for one scanclk cycle to prevent the PLL counters from being reloaded after reconfiguration.

■ When read_param is asserted, it indicates that the scan chain contents must be read and shifted out through scandataout. The bit locations and number of bits read out is dependent on the counter_type and counter_param combination. The read_param signal is sampled on the rising edge of scanclk. The read_param signal must be asserted for only one clock cycle to prevent the parameter being re-read in subsequent clock cycles. A copy of the scan chain contents are read out. The scan chain contents remain intact after a read_param.

■ The busy signal is asserted on the rising edge of scanclk following the assertion of read_param, write_param, or reconfig and remains high until the specific operation is complete. While this signal is asserted, all the scan chain inputs are ignored and the contents of the scan chain cannot be altered until it is deasserted.

For functional descriptions of the ALTPLL_RECONFIG megafunction ports, refer to the *Phase-Locked Loop Reconfiguration (ALTPLL_RECONFIG)* user guide.

The scan chain contents remain unchanged after reconfiguration. This allows multiple reconfigurations while selectively modifying one parameter each time because all other parameters retain their previous values.

PLL Dynamic Phase Shifting in the Quartus II Software

The dynamic phase shifting feature allows the output phases of individual PLL outputs to be dynamically adjusted relative to each other and to the reference clock without having to load the scan chain of the PLL. The phase is shifted by 1/8th of the VCO period at a time. The output clocks are active during this dynamic phase shift process.

Implementing PLL Dynamic Phase Shifting in the Quartus II Software

Use the ALTPLL MegaWizard interface to enable dynamic phase shifting circuitry in the ALTPLL megafunction instantiation in your design, as shown in Figure 5.

Figure 5. ALTPLL with Dynamic Phase Shifting Enabled
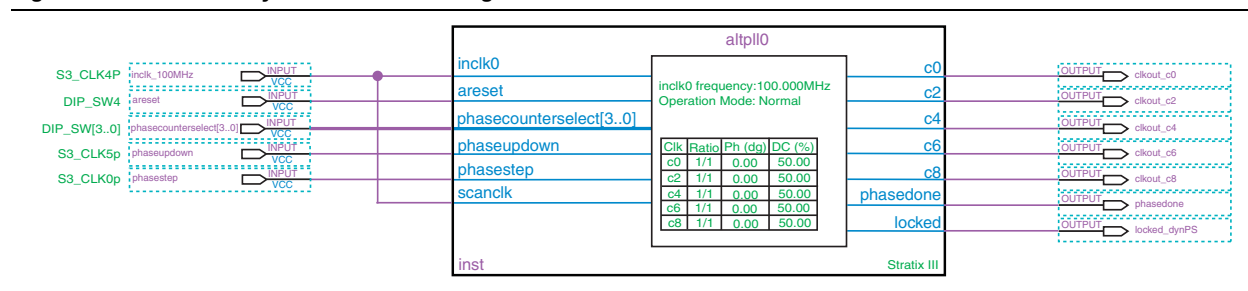
To perform one dynamic phase-shift, follow these steps:

1. Set PHASEUPDOWN and PHASECOUNTERSELECT as required.

2. Assert PHASESTEP for at least two SCANCLK cycles. Each PHASESTEP pulse allows one phase shift.

3. Deassert PHASESTEP after PHASEDONE goes low.

4. Wait for PHASEDONE to go high.

5. Repeat steps 1 through 4 as many times as required to perform multiple phase-shifts.

PHASEUPDOWN and PHASECOUNTERSELECT signals are synchronous to SCANCLK and must meet the t_{su} and t_h requirements with respect to the SCANCLK edges.

☞ You can repeat dynamic phase-shifting indefinitely. For example, in a design where the VCO frequency is set to 1,000 MHz and the output clock frequency is set to 100 MHz, performing 40 dynamic phase shifts (each one yields 125 ps phase shift) results in shifting the output clock by 180°, in other words, a phase shift of 5 ns.

Figure 6 shows the dynamic phase shifting waveform.

Figure 6. Timing Diagram for Dynamic Phase Shift
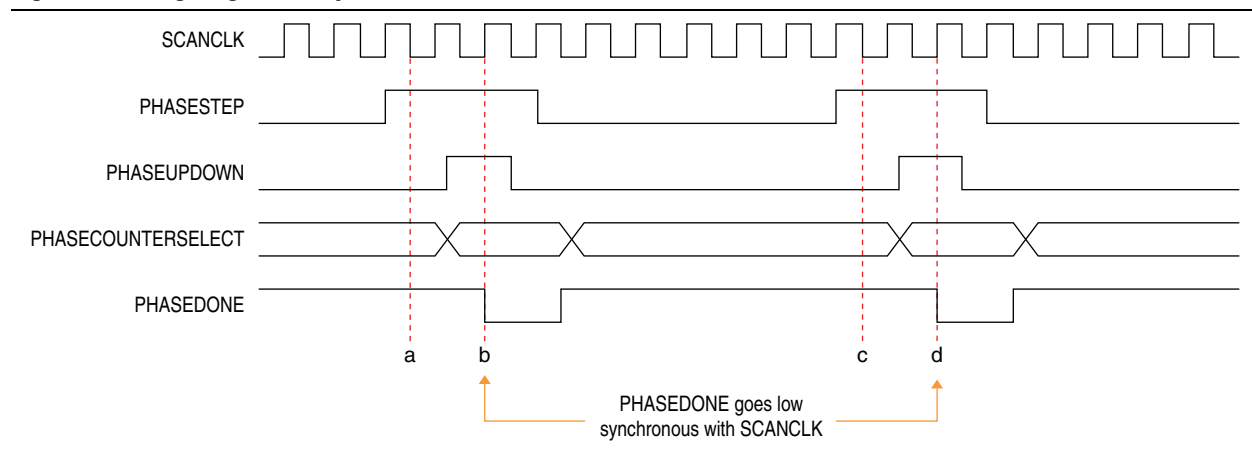
The PHASESTEP signal is latched on the negative edge of SCANCLK (a,c) and must remain asserted for at least two SCANCLK cycles. Deassert PHASESTEP after PHASEDONE goes low. On the second SCANCLK rising edge (b,d) after PHASESTEP is latched, the values of PHASEUPDOWN and PHASECOUNTERSELECT are latched and the PLL starts dynamic phase-shifting for the specified counters, and in the indicated direction. PHASEDONE is deasserted synchronous to SCANCLK at the second rising edge (b,d) and remains low until the PLL finishes dynamic phase-shifting. Depending on the VCO and SCANCLK frequencies, PHASEDONE low time may be greater than or less than one SCANCLK cycle.

You can perform another dynamic phase-shift after the PHASEDONE signal goes from low to high. Each PHASESTEP pulse enables one phase shift. PHASESTEP pulses must be at least one SCANCLK cycle apart.

PLL Reconfiguration Scan Register Bitmap

Advanced PLL users can manually select the counter and phase shift settings for Stratix III and Stratix IV devices based on information in the *Clock Networks and PLLs in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook* and the *Clock Networks and PLLs in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook,* respectively. After determining individual configuration bit settings for the different counters, and loop filter and charge pump settings, arrange the bits as shown in the bitmap in Table 2.

Table 2 provides a bitmap for the Top/Bottom PLL scan chain registers. For Left/Right PLLs, you can use the same bitmap as a reference. However, the scan chain is shorter as Left/Right PLLs only have seven output counters. The last bit shifted into the scan chain is Bit0. Bit233 is the first bit shifted in for Top/Bottom PLLs. Bit179 is the first bit shifted in for Left/Right PLLs.

Table 2. PLL Reconfiguration Scan Chain Bitmap

Bits	PLL Scan Chain Bitmap								
Bit225-Bit233	C9_mode_odd/even	C9_low_7	C9_low_6	C9_low_5	C9_low_4	C9_low_3	C9_low_2	C9_low_1	C9_low_0
Bit216-Bit224	C9_mode_bypass	C9_high_7	C9_high_6	C9_high_5	C9_high_4	C9_high_3	C9_high_2	C9_high_1	C9_high_0
Bit207-Bit215	C8_mode_odd/even	C8_low_7	C8_low_6	C8_low_5	C8_low_4	C8_low_3	C8_low_2	C8_low_1	C8_low_0
Bit198-Bit206	C8_mode_bypass	C8_high_7	C8_high_6	C8_high_5	C8_high_4	C8_high_3	C8_high_2	C8_high_1	C8_high_0
Bit189-Bit197	C7_mode_odd/even	C7_low_7	C7_low_6	C7_low_5	C7_low_4	C7_low_3	C7_low_2	C7_low_1	C7_low_0
Bit180-Bit188	C7_mode_bypass	C7_high_7	C7_high_6	C7_high_5	C7_high_4	C7_high_3	C7_high_2	C7_high_1	C7_high_0
Bit171-Bit179	C6_mode_odd/even	C6_low_7	C6_low_6	C6_low_5	C6_low_4	C6_low_3	C6_low_2	C6_low_1	C6_low_0
Bit162-Bit170	C6_mode_bypass	C6_high_7	C6_high_6	C6_high_5	C6_high_4	C6_high_3	C6_high_2	C6_high_1	C6_high_0
Bit153-Bit161	C5_mode_odd/even	C5_low_7	C5_low_6	C5_low_5	C5_low_4	C5_low_3	C5_low_2	C5_low_1	C5_low_0
Bit144-Bit152	C5_mode_bypass	C5_high_7	C5_high_6	C5_high_5	C5_high_4	C5_high_3	C5_high_2	C5_high_1	C5_high_0
Bit135-Bit143	C4_mode_odd/even	C4_low_7	C4_low_6	C4_low_5	C4_low_4	C4_low_3	C4_low_2	C4_low_1	C4_low_0
Bit126-Bit134	C4_mode_bypass	C4_high_7	C4_high_6	C4_high_5	C4_high_4	C4_high_3	C4_high_2	C4_high_1	C4_high_0
Bit117-Bit125	C3_mode_odd/even	C3_low_7	C3_low_6	C3_low_5	C3_low_4	C3_low_3	C3_low_2	C3_low_1	C3_low_0
Bit108-Bit116	C3_mode_bypass	C3_high_7	C3_high_6	C3_high_5	C3_high_4	C3_high_3	C3_high_2	C3_high_1	C3_high_0
Bit99-Bit107	C2_mode_odd/even	C2_low_7	C2_low_6	C2_low_5	C2_low_4	C2_low_3	C2_low_2	C2_low_1	C2_low_0
Bit90-Bit98	C2_mode_bypass	C2_high_7	C2_high_6	C2_high_5	C2_high_4	C2_high_3	C2_high_2	C2_high_1	C2_high_0
Bit81-Bit89	C1_mode_odd/even	C1_low_7	C1_low_6	C1_low_5	C1_low_4	C1_low_3	C1_low_2	C1_low_1	C1_low_0
Bit72-Bit80	C1_mode_bypass	C1_high_7	C1_high_6	C1_high_5	C1_high_4	C1_high_3	C1_high_2	C1_high_1	C1_high_0
Bit63-Bit71	C0_mode_odd/even	C0_low_7	C0_low_6	C0_low_5	C0_low_4	C0_low_3	C0_low_2	C0_low_1	C0_low_0
Bit54-Bit62	C0_mode_bypass	C0_high_7	C0_high_6	C0_high_5	C0_high_4	C0_high_3	C0_high_2	C0_high_1	C0_high_0
Bit45-Bit53	N_mode_odd/even	N_low_7	N_low_6	N_low_5	N_low_4	N_low_3	N_low_2	N_low_1	N_low_0
Bit36-Bit44	N_mode_bypass	N_high_7	N_high_6	N_high_5	N_high_4	N_high_3	N_high_2	N_high_1	N_high_0
Bit27-Bit35	M_mode_odd/even	M_low_7	M_low_6	M_low_5	M_low_4	M_low_3	M_low_2	M_low_1	M_low_0
Bit18-Bit26	M_mode_bypass	M_high_7	M_high_6	M_high_5	M_high_4	M_high_3	M_high_2	M_high_1	M_high_0
Bit15-Bit17	CP_2	CP_1	CP_0						
Bit10-Bit14	Reserved_4	Reserved_3	Reserved_2	Reserved_1	Reserved_0				
Bit9	VCO_postscale								
Bit4-Bit8	LF_R_4	LF_R_3	LF_R_2	LF_R_1	LF_R_0				
Bit2-Bit3	LF_C_1	LF_C_0							
Bit0-Bit1	Reserved_6	Reserved_5							

The duty cycle for each counter can be set using the high_count and low_count bits, as listed in Table 3.

Table 3. Duty Cycle Settings for Counters

Condition	Counter Bit Settings
Counter is set for odd division—ensure that counter odd division bit is set to 1	■ high_count = ($CounterDivVal$ + 1) / 2 ■ low_count = $CounterDivVal$ − high_count ■ Odd/even division bit = 1
Even nominal count is used to reconfigure M or N counters—counter bits are automatically set	■ high_count = $Nominalcount$ / 2 ■ low_count = $Nominalcount$ / 2
Odd nominal count (except 1) is used to reconfigure M or N counters—counter bits are automatically set	■ high_count = ($Nominalcount$ + 1) / 2 ■ low_count = $Nominalcount$ − high_count ■ Odd/even division bit = 1
Nominal count of 1 is used to reconfigure M or N counters—counter bits are automatically set	Bypass bit = 1

Application Examples

The following section describes two applications in which PLL reconfiguration using multiple **.mifs** and PLL dynamic phase shifting are used. The design examples provided here show the implementation details. You can download the design examples for this application note from the Altera Literature website.

Example 1: Frequency Pre-scaler in a Multi-Rate Line Card

In a multi-rate line card application, the PLL must lock to multiple frequencies to support the different data rates. The design for this application consists of a cascaded two-PLL system in which the upstream PLL acts as a frequency pre-scaler to provide the downstream PLL with a constant frequency (for example, 311 MHz). The design example for the pre-scaler is shown here. The pre-scaled frequency is achieved via the reconfiguration feature of the PLLs. Each time the frequency pre-scaler is signaled to switch its data rate, the PLL is reconfigured to the new input and output clock relationship.

Design Example 1

The frequency pre-scaler implementation shown here is for a SONET application. Sample SONET data rates are 622 MHz, 155.5 MHz, and 38.875 MHz. The frequency pre-scaler needs to output 311 MHz for all input data rates. Input signals are used to signal changes in data rates. The frequency pre-scaler example shown here uses the multiple **.mifs** for reconfiguration and the steps described in "PLL Reconfiguration: Method 2" on page 6 to achieve PLL reconfiguration.

The design consists of the ALTPLL megafunction, the ALTPLL_RECONFIG megafunction, three ROMs holding three **.mifs** corresponding to the three different input frequencies, and a state machine that selects the appropriate **.mif** and outputs the appropriate control signals to achieve dynamic reconfiguration every time the control signal to the state machine changes. The logic in the state machine automatically generates the signals necessary to achieve constant output frequency of 311 MHz.

Design Files for Stratix III Devices

Unzip **an454_prescaler_de1_SIII.zip** and compile it in the Quartus II software. The three **.mifs** have been setup with the example. Run timing simulation using the Vector Waveform File (**.vwf**) provided with the example. After mifselect changes, the PLL might lose lock after reconfiguration. After it regains lock, the PLL output frequency is always 311 MHz for different PLL input frequencies of 38.875, 155.5 or 622 MHz.

Design Files for Stratix IV Devices

Unzip **an454_prescaler_de1_SIV.zip** and compile it in the Quartus II software. This design file is similar to **an454_prescaler_de1_SIII.zip**. However, the design file for Stratix IV devices is setup for simulation using ModelSim simulator because Quartus II Simulator does not support simulations for Stratix IV devices. Run gate level simulation using the Verilog Test Bench File (**.vt**) provided in the design example.

Block Diagram

Figure 7 shows the design example for Stratix III devices in the Quartus II software. The ALTPLL megafunction is generated with dynamic reconfiguration enabled. The ALTPLL_RECONFIG megafunction is generated with ports to write from external ROM (paged **.mif**). The ports are connected as shown in Figure 7.

Figure 7. PLL Reconfiguration Using Paged .mif Capability of the ALTPLL_RECONFIG Megafunction
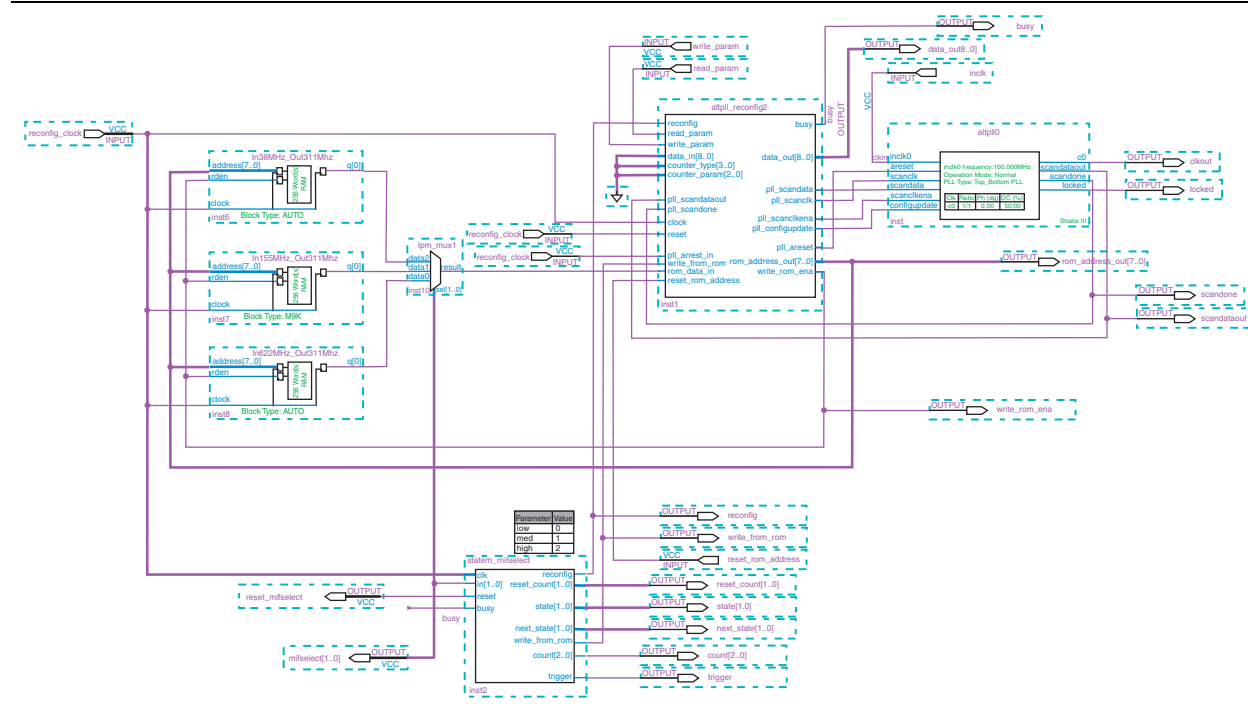
Procedure

The following steps describe the design flow for "Design Example 1" on page 11:

1. Instantiate the ALTPLL megafunction with dynamic reconfiguration enabled. Configure the PLL with one of the three data rates (which in this case is, 38.875 MHz PLL input frequency, 311 MHz output frequency). Pick the PLL counter output you require for your design. This design example uses c0 counter output. Do not quit the MegaWizard Plug-In Manager.

2. Generate a **.mif** for this configuration of the PLL using the **Generate a Configuration File button** on page 5 of the MegaWizard Plug-In Manager.

3. Change the PLL input frequency to **622 MHz** and counter output to **311 MHz** in the MegaWizard Plug-In Manager and generate the **.mif** again. Choose a different file name to avoid over-writing the file generated in step 2.

4. Change the PLL input frequency to **155 MHz** and counter output to **311 MHz** in the MegaWizard Plug-In Manager and generate a **.mif** again. Choose a different file name to avoid over-writing the file generated in step 3. You have now generated three **.mifs** for the three different input data rates for the SONET pre-scaler application.

5. Save the PLL settings in the MegaWizard Plug-In Manager. The PLL is now configured for an input frequency of 155 MHz and output frequency of 311 MHz.

6. Instantiate the ALTPLL_RECONFIG megafunction. Select **add ports to write to the scan chain from external ROM during run-time** to enable multiple **.mif** capability. Connect the ports to the ALTPLL megafunction, as shown in Figure 7.

7. Instantiate three ROMs with 1-bit wide data bus and 8-bit wide address bus. Create read enable (rden) ports for all ROMs. Initialize the three ROMs with the three **.mifs** created in steps 1 through 4 above.

   For details about ROM instantiation, refer to the *Internal Memory (RAM and ROM) User Guide*.

8. Instantiate a three-input lpm_mux and connect the ports, as shown in Figure 7.

9. Create a simple state machine that picks the appropriate ROM when input mifselect changes. Connect the ports, as shown in Figure 7. The state machine picks the ROM using select signals to the mux and outputs appropriate control signals to the ALTPLL_RECONFIG megafunction to achieve reconfiguration. Reconfiguration is possible only when the busy signal is deasserted. Any changes to mifselect when busy is high are ignored.

   Table 4 details the input signals and their effect on the outputs of the state machine.

Table 4. Design Example 1: State Machine Input and Output Signals

Mifselect	Input-Output Frequency	Counters Affected
00	38.875 MHz — 311 MHz	C0
01	155 MHz — 311 MHz	C0
10	622 MHz — 311 MHz	C0
11	No Change	None

10. Compile and simulate the design by applying inputs to mifselect. Every time mifselect changes, a different **.mif** from an external ROM is loaded into the scan chain and the PLL is reconfigured with the contents of the scan chain. The simulation waveforms for Stratix III and Stratix IV devices are shown in Figure 8 and Figure 9, respectively.

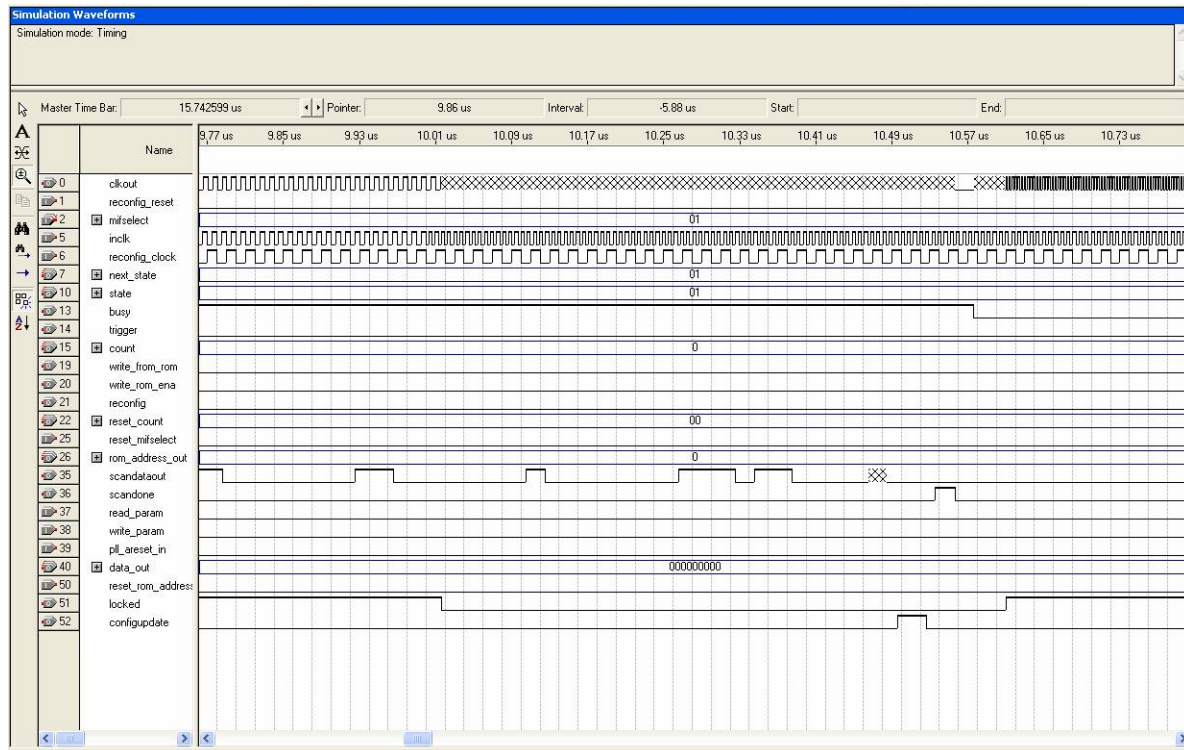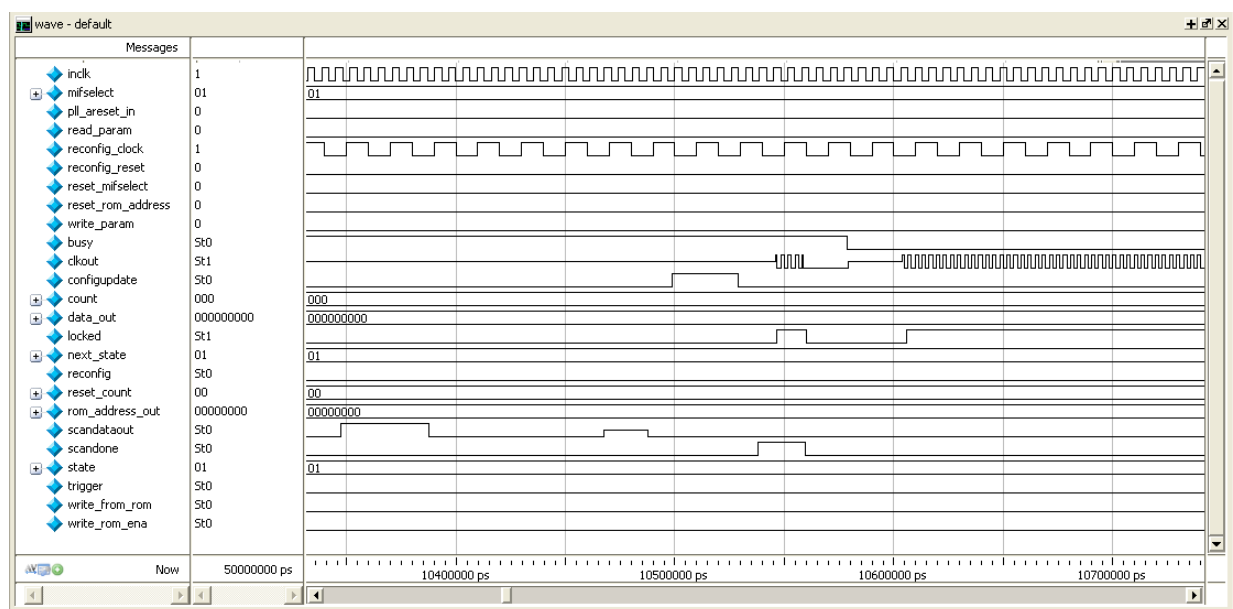**Figure 8. Design Example 1: Simulation Waveforms for Stratix III Devices**

Figure 9. Design Example 1: Simulation Waveforms for Stratix IV Devices

Example 2: Dynamic Phase Shifting Using ALTPLL Megafunction

Altera's ALTMEMPHY megafunction allows the rapid creation of a physical layer interface (PHY) in Stratix III and Stratix IV devices. The PHY safely transfers data between external memory and user logic. The Stratix III and Stratix IV ALTMEMPHY megafunction supports an initial calibration sequence to remove process variations in the FPGA and external memory device. The calibration process centers the resynchronization clock phase into the middle of the data valid window to maximize the setup and hold margin.

For details, refer to the *External Memory PHY Interface (ALTMEMPHY) (nonAFI) Megafunction User Guide*.

The auto-calibration controller, which resides in the ALTMEMPHY, uses dynamic phase shifting of the clock to determine a valid data capture window. The auto-calibration controller has a data stream and clock for inputs. This clock is fed into a reconfigurable PLL and is phase shifted in steps of 1/8th of the VCO period. The controller uses the phase shifted clock to sample the data, which is compared to the expected data. The controller shifts the phase of the clock until a valid data capture window is determined.

Design Example 2

The application described here uses dynamic phase shifting to shift the clock edge. The design example consists of a simple state machine with inputs dyn_phase and counter and the ALTPLL megafunction. The counter signal is phasecounterselect input to the PLL. You can shift the counter output phase forward or backward depending on dyn_phase. The logic in the state machine automatically generates the signals necessary to achieve phase shift.

For Stratix III and Stratix IV phasecounterselect settings, refer to the *Clock Networks and PLLs in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook* and the *Clock Networks and PLLs in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook,* respectively*.*

Design Files for Stratix III Devices

Unzip **an454_altpll_dynphase_de2_SIII.zip** and compile it in the Quartus II software. Run timing simulation using the **.vwf** file provided with the example. The example is set up to step the phase up two times. You can observe the change in the PLL counter output. The edge gets shifted forward by 1/8th VCO period after the first phase shift and by 1/4th VCO period after the second phase shift.

Design Files for Stratix IV Devices

Unzip **an454_altpll_dynphase_de2_SIV.zip** and compile it in the Quartus II software. This design is similar to **an454_altpll_dynphase_de2_SIII.zip.** However, the design file for Stratix IV devices is setup for simulation using ModelSim simulator because Quartus II Simulator does not support simulations for Stratix IV devices. Run gate level simulation using the Verilog Test Bench File (**.vt**) provided in the design example.

Block Diagram

Figure 10 shows the design example for Stratix III devices in the Quartus II software. The ALTPLL megafunction is generated with dynamic phase shift reconfiguration enabled. The dynamic phase shift control ports are connected to the state machine as shown. This example targets all counters of the PLL.

Figure 10. Dynamic Phase Shift

Parameter	Value
phase_up	B"01"
phase_down	B"10"
high	B"00"
up	B"1"
down	B"0"
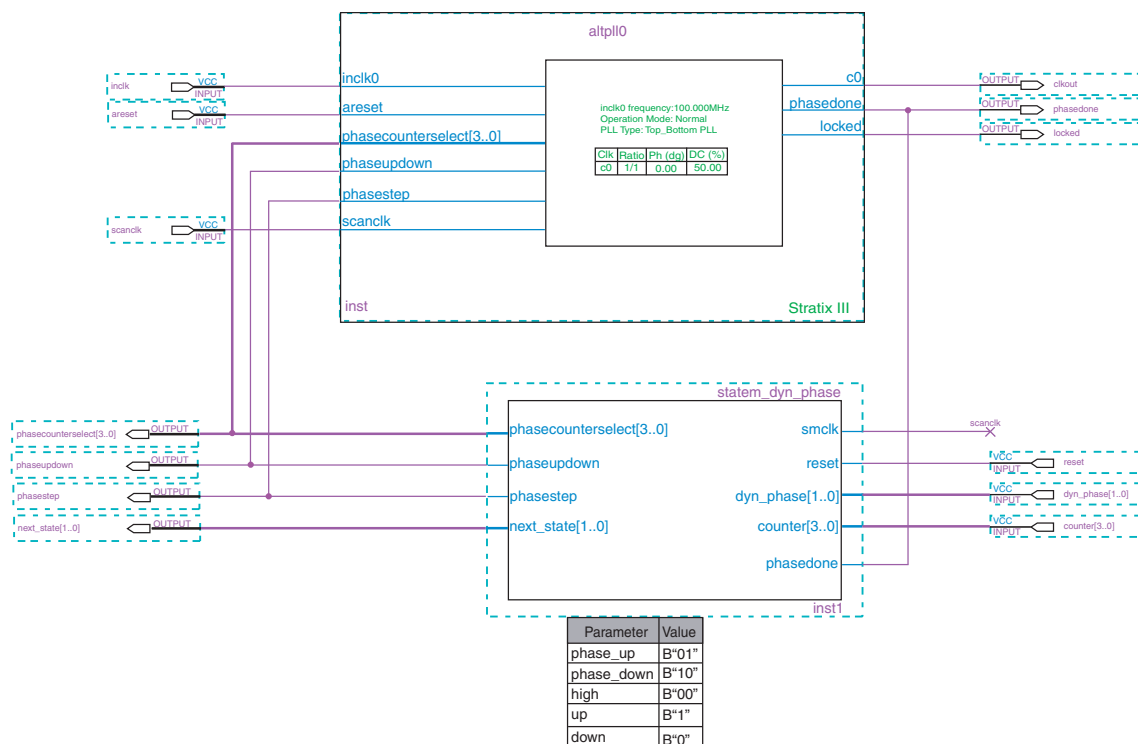
Procedure

The following steps describe the design flow for "Design Example 2" on page 15:

1. Instantiate the ALTPLL megafunction with dynamic phase shifting enabled. Configure the PLL with the desired input and output frequencies using the megafunction.

2. Create the state machine with inputs and outputs, as shown in Figure 10. For details about input signals and their effect on the outputs of the state machine, refer to Table 5. When dyn_phase is 10, the state machine outputs the appropriate signals to shift the phase forward. When dyn_phase is 11, the state machine outputs the appropriate signals to shift the phase backward. For details about how to set up these output signals, refer to the timing diagram in Figure 6 on page 9.

Table 5. Design Example 2: State Machine Input Ports and the Effect on the Outputs

Counter	dyn_phase	Phase	Counters Affected
0	10	1 step forward	All PLL counters
0	11	1 step backward	All PLL counters
0	00	No change	None
0	01	No change	None

3. Compile and simulate the design by applying inputs to the counter and dyn_phase. Any change to dyn_phase must be made only when phasedone is asserted. All changes to dyn_phase when phasedone is low are ignored. Figure 11 and Figure 12 show the simulation waveforms for Stratix III and Stratix IV devices, respectively.

4. PLL counter c0 is shown in this design example, but the counter input to the state machine targets all PLL output counters. You can change the counter input to the state machine to affect specific PLL output counters.

> For Stratix III and Stratix IV phasecounterselect settings, refer to the *Clock Networks and PLLs in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook* and the *Clock Networks and PLLs in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook.*

Figure 11. Design Example 2: Dynamic Phase Shift Simulation Output for Stratix III Devices
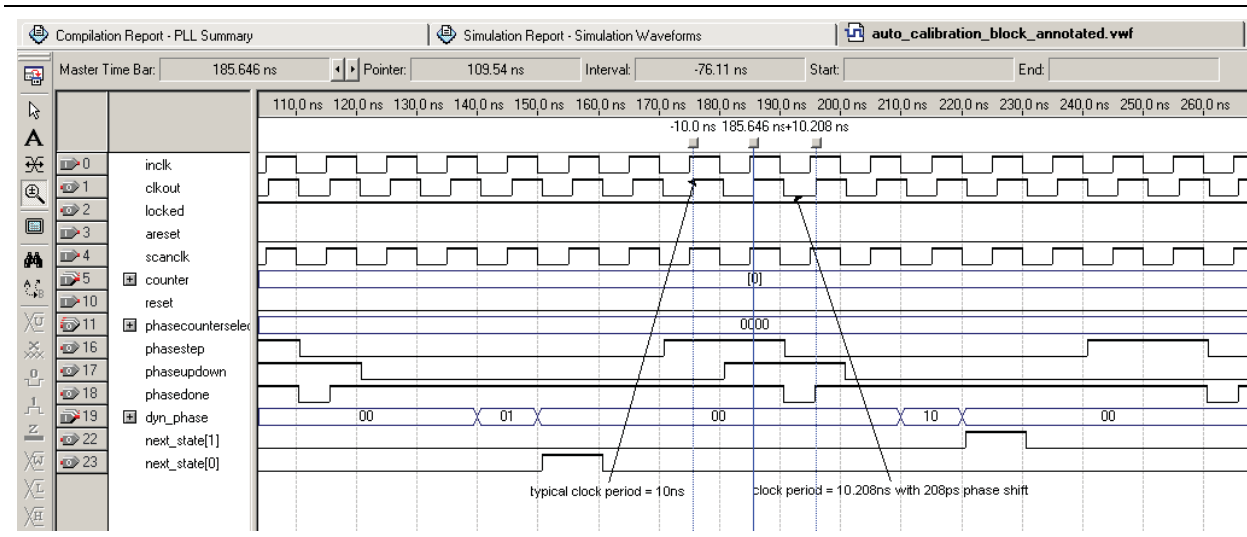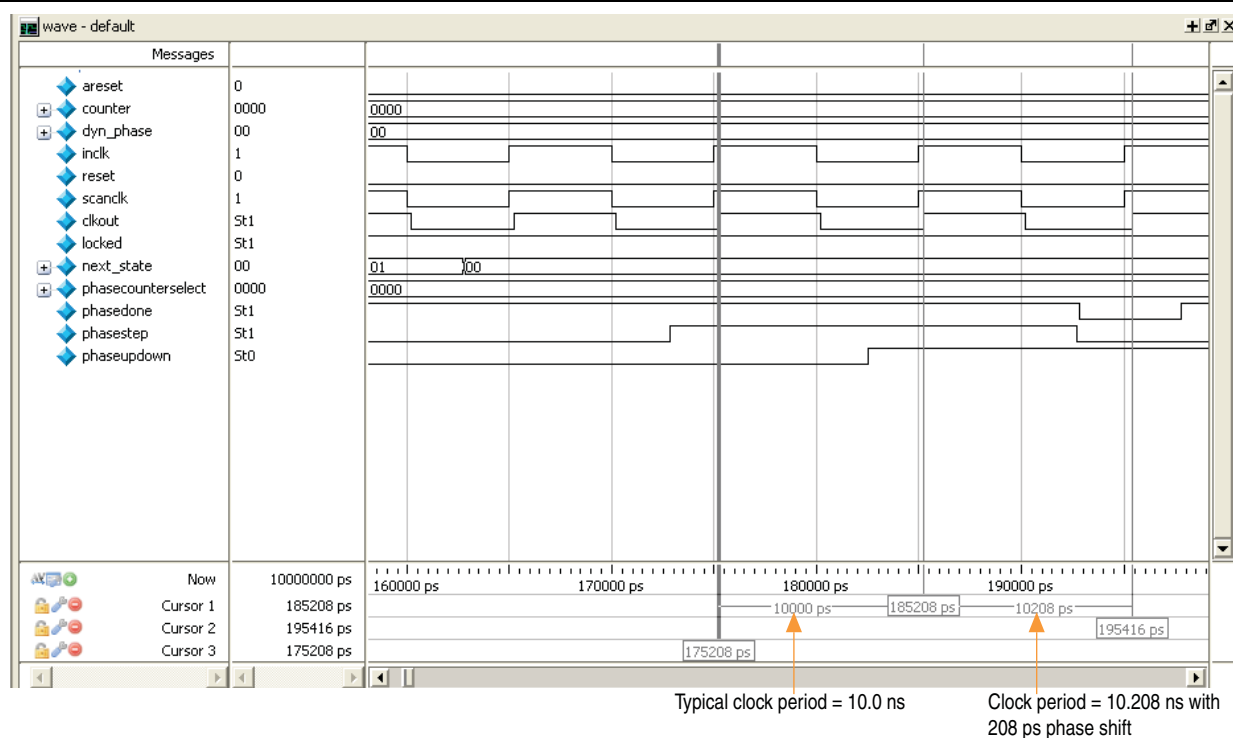
Figure 12. Design Example 2: Dynamic Phase Shift Simulation Output for Stratix IV Devices

Typical clock period = 10.0 ns

Clock period = 10.208 ns with 208 ps phase shift

Another application in which dynamic phase shifting is useful is in Nios® II applications. When using the Nios II processor, off-chip memory interfacing is necessary for most applications. Improper clocking of memory devices can cause issues such as not being able to run code or perform back-to-back transactions in memory. You can use dynamic phase shifting to tune the PLL to determine the valid signal window for accessing the memory device. The hardware can be created in the SOPC builder to contain the necessary peripherals to read the PLL parameters from the ALTPLL megafunction. The software which is part of the design can be created as a Nios IDE project written in C to shift the off-chip memory clock output of the PLL, test the SDRAM memory, and calculate the boundaries and center of the SDRAM valid signal window.

Design Considerations

Consider the following information when using PLL reconfiguration:

■ Changing pre-scale and feedback counter settings (m, n), charge pump/loop filter settings affects the PLL VCO frequency, which might require the PLL to relock to the reference clock.

■ Changing the m counter phase shift setting changes the phase relationship of the output clocks with respect to the reference clock, which also requires the PLL to relock. Although the exact effect of changing pre-scale and feedback counter settings (m, n) depends on what setting is changed, any change typically requires relocking.

- If you choose not to use a **.mif** to specify the initial contents of the scan chain, the scan chain is blank when the device enters user mode. The PLL is configured to its initial state based on the programmed settings in the Programmer Object File (**.pof**).

- Adding phase shift using the fm phase shift setting pulls in all the PLL clock outputs with respect to reference clock, effectively adding a negative phase shift. This is because fm is in the feedback path.

- When making changes to the loop elements (*m*, *n*, *m* counter phase, Icp, R, C), Altera recommends disabling PLL outputs to the logic array using the `clkena` signals available on the ALTCLKCTRL megafunction. This eliminates the possibility of an overfrequency condition affecting system logic while PLL is regaining lock.

- Changes to post-scale counters (C) and phase do not affect the PLL lock status or VCO frequency. The resolution of phase shift is always a function of VCO frequency, with the smallest incremental step equal to 1/8th of the VCO period.

- When the phase relationship between output clocks is important, Altera recommends resynchronizing the PLL using the `areset` signal. This resets all internal PLL counters and re-initiates the locking process.

- The Stratix III and Stratix IV PLL scan chain supports a free running `scanclk`, so there is no need to precisely control the start and stop of the clock.

- Changes to the *m* or *n* counter values affect all of the output clock frequencies. Output counters can also be individually reconfigured.

- The `scandone` signal is synchronous with the positive edge of `scanclk` and is asserted for at least one clock cycle by the ALTPLL megafunction after reconfiguration is complete.

Conclusion

PLL reconfiguration is a powerful feature that system designers can use to vary the PLL clock output frequency and to phase shift at any stage. Important considerations, such as loss of lock, glitches, and output phase relationships might affect your selections of the PLL counter and phase shift settings. PLL reconfiguration time is typically less than 20 μs, allowing you to switch rapidly between operating modes. The flexibility offered by the Stratix III and Stratix IV PLL makes it a superior clock management system.

Document Revision History

Table 6 lists the revision history for this document.

Table 6. Document Revision History (Part 1 of 2)

Date	Version	Changes
August 2012	3.2	■ Minor text edits to "PLL Dynamic Phase Shifting in the Quartus II Software".
July 2012	3.1	■ Updated "Conclusion" section. ■ Updated design file for Stratix IV devices in Design Example 1.

Table 6. Document Revision History (Part 2 of 2)

Date	Version	Changes
October 2011	3.0	■ Added steps to perform dynamic phase-shift to "Implementing PLL Dynamic Phase Shifting in the Quartus II Software" ■ Updated Figure 6 ■ Replaced text in "PLL Reconfiguration Scan Register Bitmap" with Table 3
December 2009	2.0	■ Updated the document to include Stratix IV information ■ Added simulation waveforms for Stratix IV devices
May 2009	1.2	■ Updated the "PLL Reconfiguration: Method 1" section ■ Replaced Figure 3 and Figure 4 ■ Updated the "Reconfiguration Signals" section ■ Updated Table 2
October 2007	1.1	Changed Figure 4.
October 2007	1.0	Initial Release