Implementing PLL Reconfiguration in Cyclone III Devices

AN507-2.0 Application Note

This application note describes the flow for implementing phase-locked loop (PLL) reconfiguration in Cyclone® III devices and how to use the PLL reconfiguration feature. Use this application note in conjunction with the following literature:

- *Clock Networks and PLLs in Cyclone III Devices* chapter in volume 1 of the *Cyclone III Device Handbook*
- *Phase-Locked Loop (ALTPLL) Megafunction User Guide*
- *Phase-Locked Loop Reconfiguration (ALTPLL_RECONFIG) Megafunction* user guide

This application note discusses the following topics:

- Overview of PLL reconfiguration in Cyclone III devices
- Complete flow on implementing real-time PLL reconfiguration feature in a frequency prescaler application
- Complete flow on implementing dynamic phase-shifting feature
- Design considerations that you must consider when selecting PLL parameters for reconfiguration

Overview

PLLs use several divide counters and different voltage-controlled oscillator taps to perform frequency synthesis and phase shifts. In Cyclone III PLLs, you can reconfigure the counter settings and dynamically shift the phase of 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 adjust the clock output phase at any stage of the design. For example, 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 for this example allows you to switch between the two output frequencies within a few microseconds.

You can also use reconfiguration to adjust clock-to-out (t_{CO}) delays in real time by shifting the phase of the output clock. This approach eliminates the need to regenerate a configuration file with the new PLL settings.

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

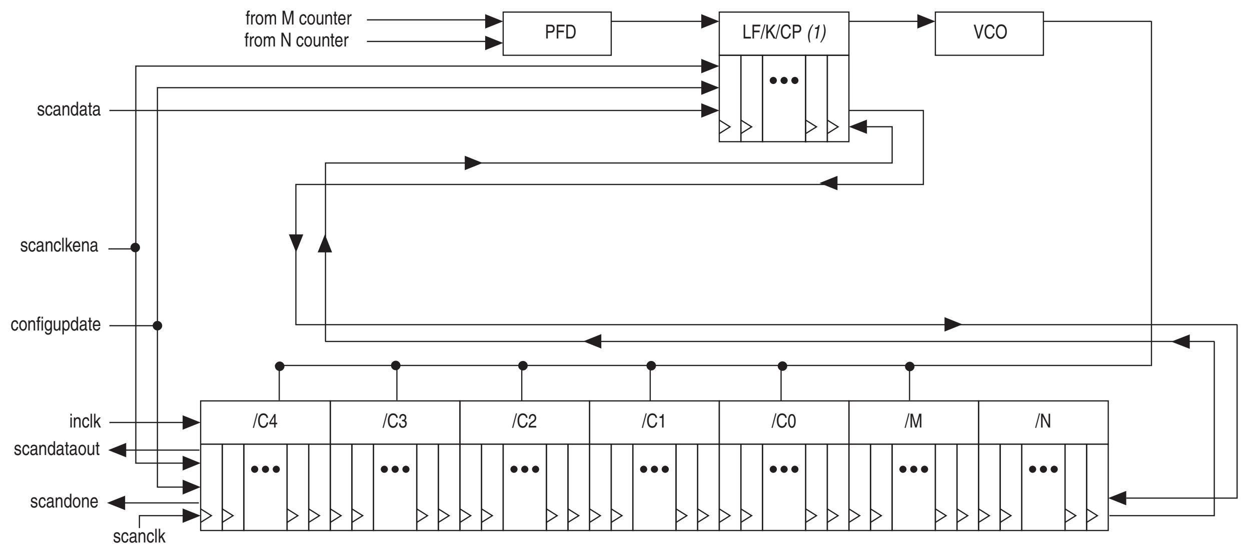
Subscribe

The following PLL components are reconfigurable in real time:

- Prescale counter (N)
- Feedback counter (M)
- Post-scale counters (C0-C4)
- Post-divider (K)
- Charge pump current (I_{CP}) and loop filter components (R, C)

Figure 1 shows how PLL counter settings can be adjusted dynamically by shifting their new settings into a serial shift-register chain or scan chain. Serial data is fed into the scan chain via the `scandata` port and the shift registers are clocked by the `scanclk` port. 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 in the hexadecimal (**.hex**) file or memory initialization file (.mif) format. Refer to "PLL Reconfiguration Scan Register Bitmap" on page 7 for details on memory initialization file settings.

Figure 1. PLL Reconfiguration Scan Chain



Note to Figure 1:

(1) 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 voltage-controlled oscillator (VCO).

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

Implementing PLL Reconfiguration using Quartus II Software

You can use the Quartus® II ALTPLL MegaWizard® Plug-in Manager to enable the reconfiguration circuitry in the ALTPLL megafunction instantiation in your design. The ALTPLL_RECONFIG megafunction simplifies the process of reconfiguring Cyclone III 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*.

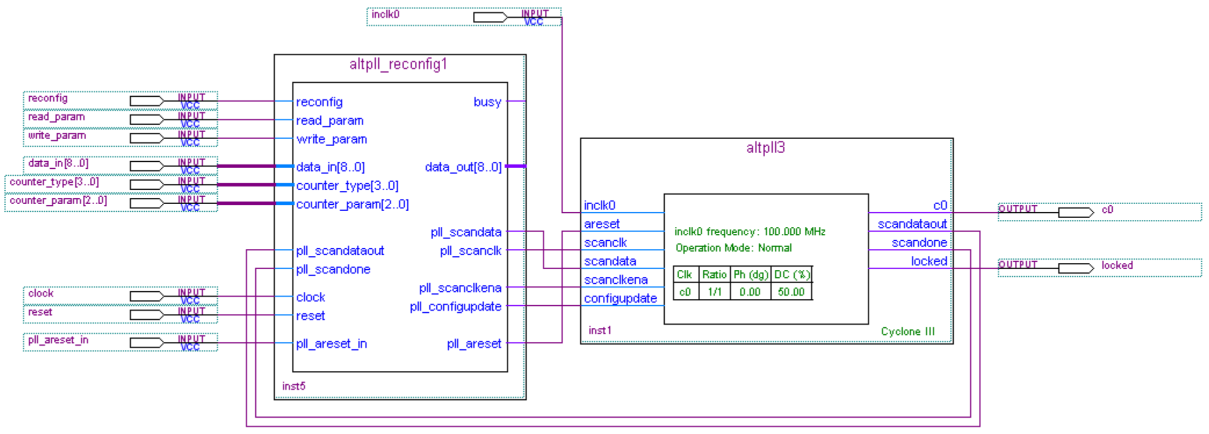
The following two methods describe how PLLs can be reconfigured with the ALTPLL_RECONFIG megafunction. Either method may be used for reconfiguration.

- Method 1 is used to reconfigure a specific PLL counter without affecting other PLL settings. This method is used when one or a few PLL settings need to be modified at a time.
- Method 2 is used to reconfigure all the PLL counter settings using a memory initialization file. The scan chain is updated with the memory initialization file in .mif or .hex file format and the PLL counters are updated with the content of the scan chain.

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.

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



The following steps describe PLL reconfiguration using Method 1:

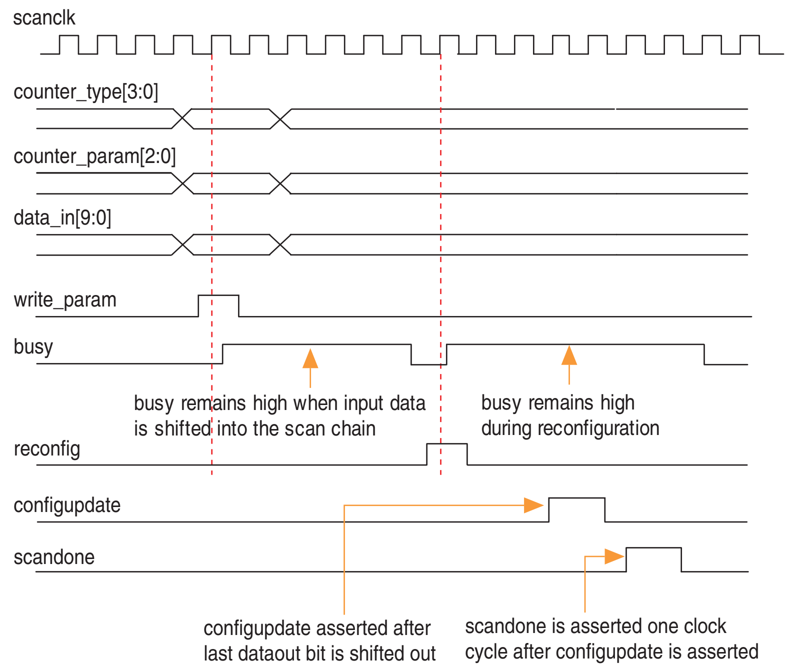
1. With the 4-bit `counter_type` port, specify a counter type (`C0-C4`, `M`, `N`, `CP/LF`, `VCO`).
2. With the 3-bit `counter_param` port, specify which parameter should be updated for the counter type in step 1.

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

3. With the 9-bit `data_in` port, specify a value for the `counter_param` in step 2.
4. Assert the `write_param` signal for one `scanclk` cycle to allow the inputs from steps 1 through 3 to be written to the scan chain.
5. The `busy` signal is asserted as soon as the `write_param` signal is asserted. 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 deasserted, you can write a different `counter_type` into the scan chain following steps 1 through 5.
7. When all data is shifted into the scan chain, assert the `reconfig` signal after the `busy` signal is deasserted to reconfigure the PLL with the content of the scan chain.

The timing diagram shown in Figure 3 illustrates the relationship between the `counter_type`, `counter_param`, `datain`, `reconfig`, and `busy` signals to achieve PLL reconfiguration through the ALTPLL_RECONFIG megafunction.

Figure 3. Timing Diagram for PLL Reconfiguration Using Method 1


scanclk
counter_type[3:0]
counter_param[2:0]
data_in[9:0]
write_param
busy
busy remains high when input data is shifted into the scan chain
busy remains high during reconfiguration
reconfig
configupdate
scandone
configupdate asserted after last dataout bit is shifted out
scandone is asserted one clock cycle after configupdate is asserted

PLL Reconfiguration: Method 2

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

The design set-up is similar to Method 1, except that the `counter_type`, `counter_param`, and `datain` ports are unused 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 make sure the .mif or .hex file 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 Configuration File** in the ALTPLL MegaWizard. Use the generated .mif or .hex file for reconfiguration in your design.

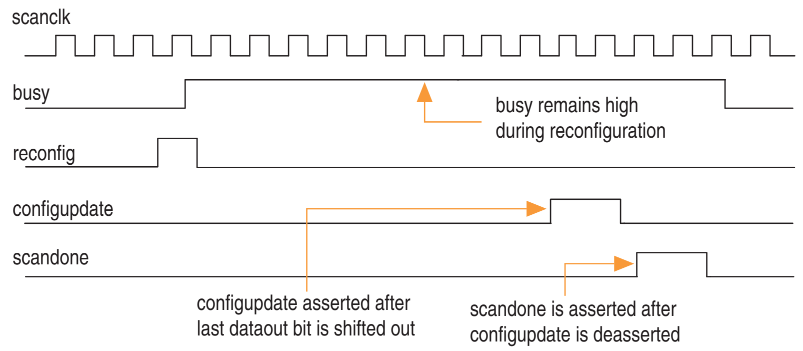
For Cyclone III devices, multiple reconfigurations are possible using multiple .mif or .hex files. You can create multiple .mif or .hex files with the MegaWizard without having to recompile your design. Refer to "Application Example 1" on page 10 for details.

The following steps describe PLL reconfiguration using Method 2:

1. Create the .mif or .hex file for the new configuration settings.
2. Associate the ALTPLL_RECONFIG megafunction with the .mif or .hex file.
3. Assert the `reconfig` signal to update the scan chain and the PLL counters with the contents of the .mif or .hex file.

Figure 4 shows the timing diagram for reconfiguration using a .mif or .hex file in the ALTPLL_RECONFIG megafunction.

Figure 4. Timing Diagram for Reconfiguration Using a .mif or .hex File in the ALTPLL_RECONFIG Megafunction


scanclk
busy
busy remains high during reconfiguration
reconfig
configupdate
scandone
configupdate asserted after last dataout bit is shifted out
scandone is asserted after configupdate is deasserted

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`. The `reconfig` signal needs to be asserted only for one `scanclk` cycle to prevent the PLL counters from being reloaded after reconfiguration.

- When the `read_param` signal is asserted, it indicates that the content of the scan chain should be read and shifted out through the `scandataout` port. The bit locations and number of bits read out are dependent on the combination of `counter_type` and `counter_param`. 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 from being re-read in subsequent clock cycles. A copy of the scan chain content is read out. The scan chain content remains intact after a `read_param` operation.
- The `busy` signal is asserted when the `read_param`, `write_param`, or `reconfig` operation is asserted and remains high until the specific operation is complete. While this signal is asserted, all scan chain inputs are ignored and the content 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) Megafunction* user guide.

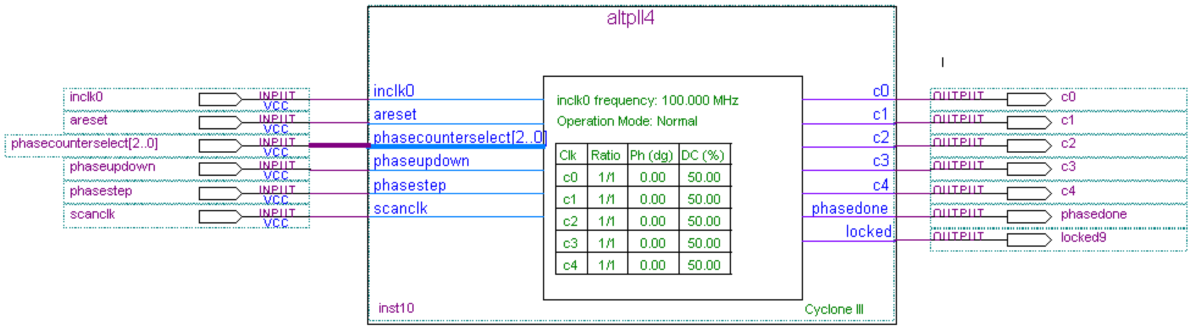
The content of the scan chain remains unchanged after reconfiguration. This allows for multiple reconfigurations while selectively modifying one parameter during each configuration because all other parameters retain their previous values.

Implementing 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 period of the voltage-controlled oscillator (VCO) at a time. The output clocks are active during this dynamic phase-shift process.

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

Figure 5. ALTPLL Megafunction with Dynamic Phase Shifting Enabled



inclk0 frequency: 100.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00
c1	1/1	0.00	50.00
c2	1/1	0.00	50.00
c3	1/1	0.00	50.00
c4	1/1	0.00	50.00

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.

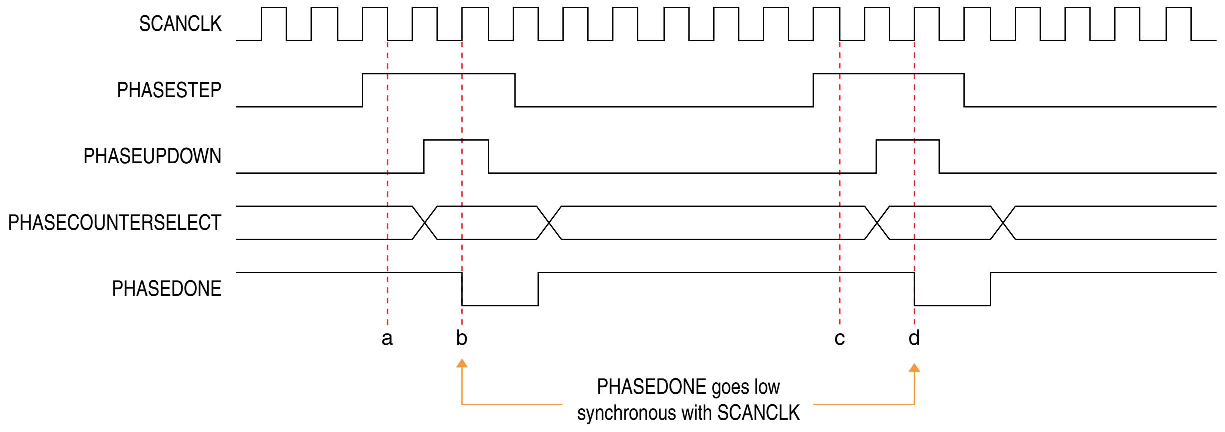
3. Deassert PHASESTEP after PHASEDONE goes low.
4. Wait for PHASEDONE to go high.
5. Repeat steps 1 through 4 as many times as required to perform multiple phase-shifts.

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 based on the information found in the *Clock Networks and PLLs in Cyclone III Devices* chapter in volume 1 of the *Cyclone III Device Handbook*. 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 on page 9.

Table 2 on page 9 provides a bitmap for the PLL scan chain registers. The last bit shifted into the scan chain is `Bit0`. `Bit143` is the first bit shifted in. The duty cycle for each counter can be set using the `high_count` and `low_count` bits.

The duty cycle for each counter can be set using the `high_count` and `low_count` bits, as listed in Table 1.

Table 1. Duty Cycle Settings for Counters

Condition	Counter Bit Settings
To generate duty cycle of 50% and an odd C counter	■ `high_count` = (*CounterValue* + 1) / 2 ■ `low_count` = *CounterValue* – `high_count` ■ Odd/even division bit = 1
To generate duty cycle of 50% and an even C counter	■ `high_count` = *CounterValue* / 2 ■ `low_count` = *CounterValue* / 2 ■ Odd/even division bit = 0
To generate duty cycle of 50% and a C counter value of 5	■ `high_count` = 3 ■ `low_count` = 2 ■ 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

Table 2. PLL Reconfiguration Scan Chain Bitmap

Bits	PLL Scan Chain Bitmap								
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	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
Bit36-Bit44	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
Bit27-Bit35	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
Bit18-Bit26	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
Bit15-Bit17	CP_2	CP_1	CP_0	—	—	—	—	—	—
Bit10-Bit14	Reserved_0	Reserved_1	Reserved_2	Reserved_3	Reserved_4	—	—	—	—
Bit9	VCO_postscale_0	—	—	—	—	—	—	—	—
Bit4-Bit8	LF_R_0	LF_R_1	LF_R_2	LF_R_3	LF_R_4	—	—	—	—
Bit2-Bit3	LF_C_0	LF_C_1	—	—	—	—	—	—	—
Bit0-Bit1	Reserved_5	Reserved_6	—	—	—	—	—	—	—

Application Example 1

This section and the following one describe two applications where the PLL is reconfigured using multiple .mif files and PLL dynamic phase shifting. The design examples provided show the implementation details. You can download the design examples from the Application Notes web page at www.altera.com/literature/lit-an.jsp.

PLL Reconfiguration in a Display Application

In a display application, the Cyclone III PLL must lock to multiple frequencies to support multiple refresh rates. The multiple refresh rates can be achieved through the reconfiguration feature of the Cyclone III PLL. Every time the application needs to generate a new refresh rate, the PLL is reconfigured to the new input and output clock relationship.

Design Example 1

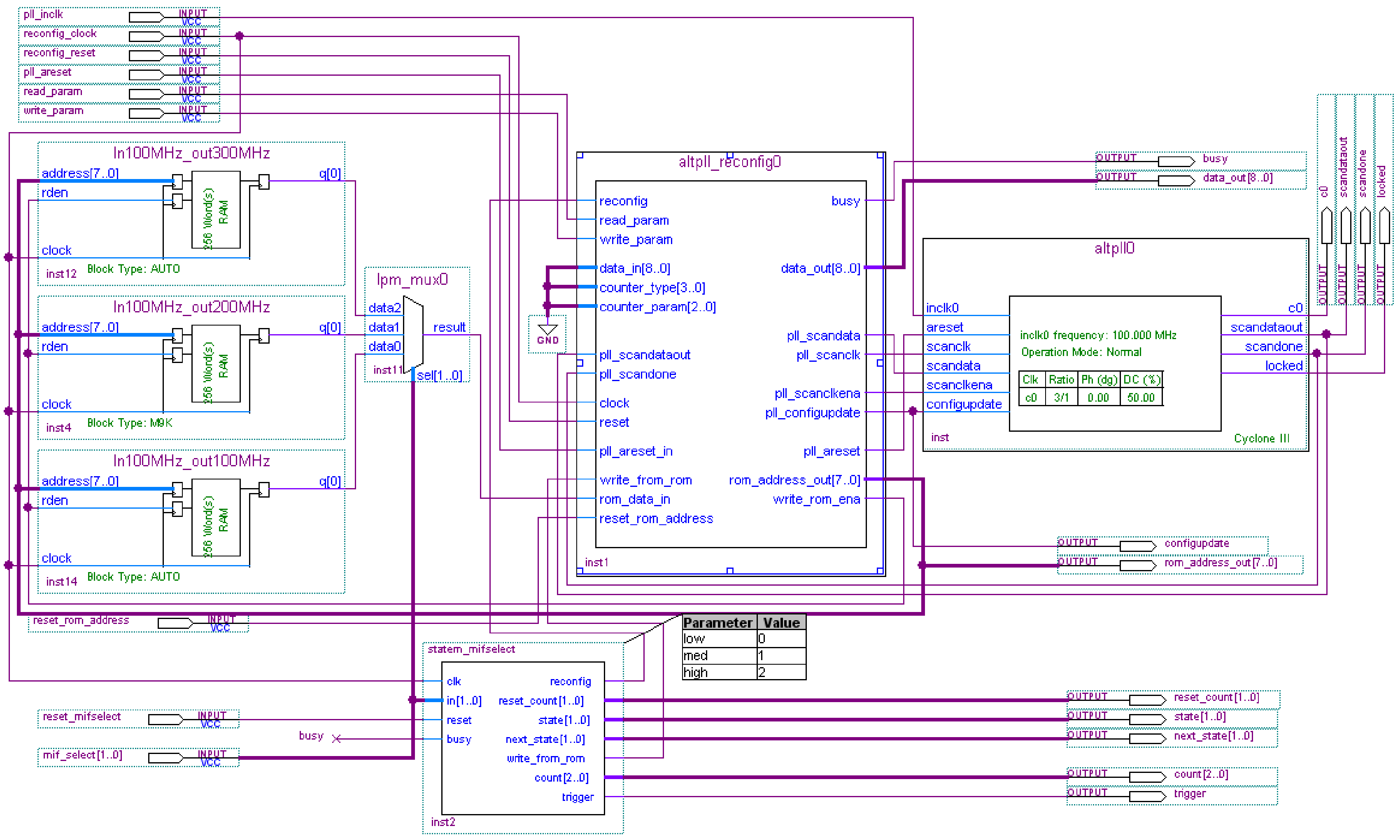
This design example uses multiple **.mif** files to reconfigure the PLL. Perform the steps described in "PLL Reconfiguration: Method 2" on page 4 to achieve PLL reconfiguration.

The design uses the ALTPLL and ALTPLL_RECONFIG megafunctions, three ROMs holding three **.mif** files corresponding to the three different input frequencies, and a state machine that selects the appropriate **.mif** file and generates 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.

Block Diagram

Figure 7 on page 11 shows the design example in the Quartus II software. The ALTPLL megafunction is generated with dynamic reconfiguration enabled and with ports to write from an external ROM (paged .mif file).The ports are connected as shown in Figure 7 on page 11.

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



Parameter	Value
low	0
med	1
high	2

Procedure

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

1. Instantiate the ALTPLL megafunction with dynamic reconfiguration enabled. Configure the PLL with one of the three data rates (which, in this case, is 100 MHz of PLL input frequency and 100 MHz of output frequency). Pick the PLL counter output you require for your design. This design example uses the C0 counter output. Do not quit the MegaWizard.
2. Generate a .mif file for this configuration of the PLL using the Generate a Configuration File button on page 5 of the MegaWizard.
3. Change the PLL counter output to 200 MHz in the MegaWizard and generate a .mif file again. Choose a different file name to avoid overwriting the file generated in step 2.
4. Change the PLL counter output to 300 MHz in the MegaWizard and generate a .mif file again. Choose a different file name to avoid overwriting the file generated in step 3. You have now generated three .mif files for the three different input data rates of the display application.
5. Save the PLL settings in the MegaWizard. The PLL is now configured for an input frequency of 100 MHz and output frequency of 300 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 on page 11.
7. Instantiate three ROMs with a 1-bit wide data bus and an 8-bit wide address bus. Create read enable (`rden`) ports for all ROMs. Initialize the three ROMs with the three .mif files created in steps 1 through 4 above.

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

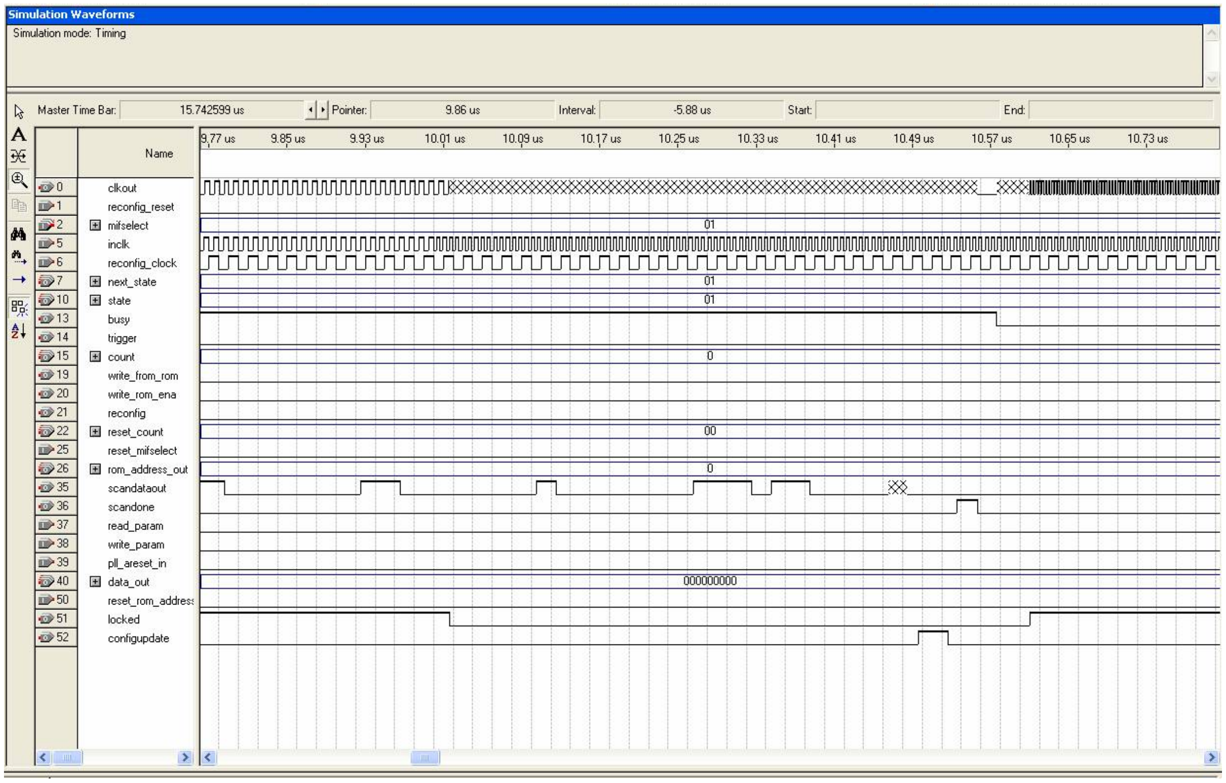
8. Instantiate a three-input `lpm_mux` and connect the ports, as shown in Figure 7 on page 11.
9. Create a simple state machine that picks the appropriate ROM when the input to the `mifselect` port changes. Connect the ports, as shown in Figure 7 on page 11. To achieve reconfiguration, the state machine chooses the ROM using select signals to the multiplexer and generates the appropriate control signals to the ALTPLL_RECONFIG megafunction. Reconfiguration is possible only when the `busy` signal is deasserted. Any changes to `mifselect` when the `busy` signal is high are ignored. Table 3 shows the input signals and their effects on the outputs of the state machine.

Table 3. Design Example 1: State Machine Input and Output Signals

mifselect	Input-Output Frequency	Counters Affected
00	100 - 100 MHz	C0
01	100 - 200 MHz	C0
10	100 - 300 MHz	C0
11	No Change	None

10. Compile and simulate the design, applying different inputs to `mifselect`. Every time `mifselect` changes, a different .mif file from an external ROM is loaded into the scan chain and the PLL is reconfigured with the content of the scan chain. Figure 8 shows the simulation waveforms.

Figure 8. Design Example 1: Simulation Waveforms



Simulation Waveforms

Simulation mode: Timing

Master Time Bar: 15.742599 us Pointer: 9.86 us Interval: -5.88 us Start: End:

Application Example 2

Dynamic Phase Shifting Using ALTPLL Megafunction

The ALTMEMPHY megafunction allows for the rapid creation of a physical layer interface (PHY) in Cyclone III devices. The PHY safely transfers data between the external memory and user logic. The Cyclone III 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 valid data window to maximize the set-up and hold margin.

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

The autocalibration controller, which resides in the ALTMEMPHY megafunction, uses the dynamic phase shifting of the clock to determine a valid data capture window. The autocalibration controller has a data stream and clock for inputs. This clock is fed into a reconfigurable PLL and 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

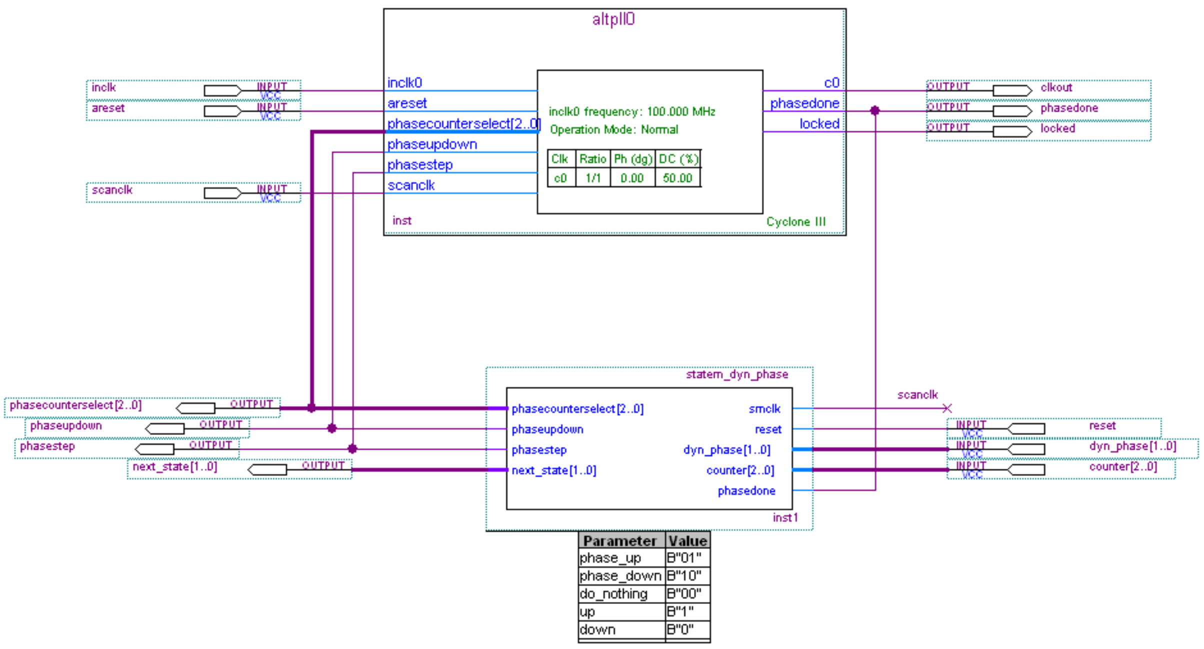
The application described here uses dynamic phase shifting to shift the clock edge. The design example consists of a simple state machine with input signals `dyn_phase` and `counter`, and the ALTPLL megafunction. The `counter` signal is the input to `phasecounterselect` of the PLL. You can shift the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

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

Block Diagram

Figure 9 on page 14 shows the design example 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 in Figure 9 on page 14. This example targets all counters of the PLL.

Figure 9. Cyclone III Dynamic Phase Shift



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

Procedure

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

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 9 on page 14. Refer to Table 4 for details on input signals and their effects on the outputs of the state machine. When `dyn_phase` is `10`, the state machine generates the appropriate output signals to shift the phase forward. When `dyn_phase` is `11`, the state machine generates the appropriate output signals to shift the phase backward. For details on how to set up these output signals, refer to the timing diagram in Figure 6 on page 7.

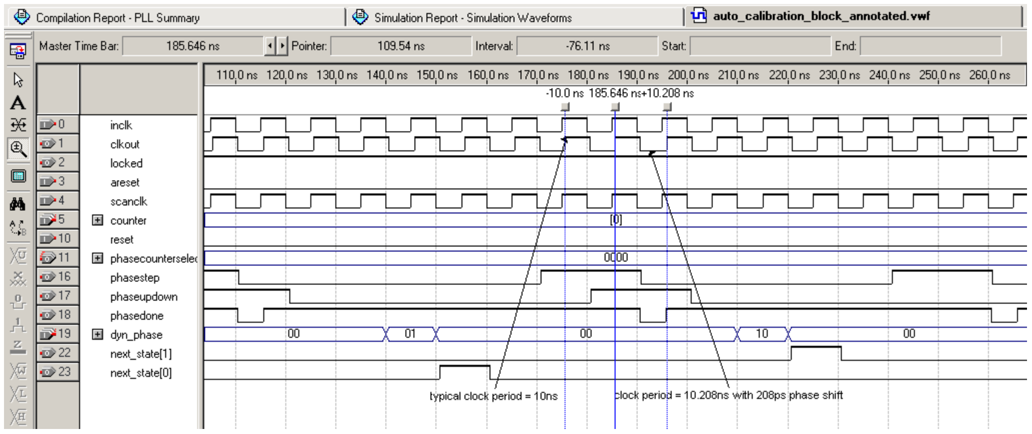
   Table 4 details the input signals and their effects on the outputs of the state machine.

Table 4. Design Example 2: State Machine Input Ports and Effects on Output

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, applying inputs to `counter` and `dyn_phase`. Any change to `dyn_phase` should be made only when `phasedone` is asserted. All changes to `dyn_phase` when `phasedone` is low are ignored. Figure 10 shows the simulation waveforms.

Figure 10. Design Example 2: Dynamic Phase-Shift Simulation Output


Compilation Report - PLL Summary
Simulation Report - Simulation Waveforms
auto_calibration_block_annotated.vwf
Master Time Bar: 185.646 ns
Pointer: 109.54 ns
Interval: -76.11 ns
Start:
End:
110,0 ns 120,0 ns 130,0 ns 140,0 ns 150,0 ns 160,0 ns 170,0 ns 180,0 ns 190,0 ns 200,0 ns 210,0 ns 220,0 ns 230,0 ns 240,0 ns 250,0 ns 260,0 ns
-10.0 ns 185.646 ns+10.208 ns
0 inclk
1 clkout
2 locked
3 areset
4 scanclk
5 counter
10 reset
11 phasecounterselec
16 phasestep
17 phaseupdown
18 phasedone
19 dyn_phase
22 next_state[1]
23 next_state[0]
[0]
0000
00
01
00
10
00
typical clock period = 10ns
clock period = 10.208ns with 208ps phase shift

4. PLL counter C0 is shown in this design example, but the `counter` input to the state machine targets all PLL output counters. You may change the `counter` input to the state machine to affect specific PLL output counters.

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

Design Considerations

Dynamic phase shifting is also useful in Nios® II applications. When the Nios II processor is used, 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 include the necessary peripherals to read PLL parameters from the ALTPLL megafunction. The software part of the design can be created as a Nios IDE project, written in the C language, to shift the off-chip memory clock output of the PLL, test the SDRAM memory, and calculate the boundaries and center of the valid signal window for the SDRAM.

You must consider the following information when reconfiguring the PLL:

- Changing the prescale and feedback counter settings (M, N), charge pump settings, or 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 prescale 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** file 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**).
- When making changes to the loop elements (M, N, M counter phase, I_{CP}, 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 the 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 a phase shift is always a function of the VCO frequency, with the smallest incremental step equalling 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 Cyclone III 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 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 once reconfiguration is complete.

Design Examples

You can download the design examples from the Application Notes web page at www.altera.com/literature/lit-an.jsp. The following sections show how you can download and use the design examples.

Design Example 1: PLL Reconfiguration in a Display Application

Unzip Design Example 1 (an507_display_de1.zip) and compile it in the Quartus II software. The three **.mif** files have been set up with the example. Run a timing simulation using the Vector Waveform File (**.vwf**) provided with the example. After `mifselect` changes, the PLL may lose lock after reconfiguration. Once it regains lock, the PLL output frequency changes according to the **.mif** file that is selected for reconfiguration.

Design Example 2: Dynamic Phase Shifting Using the ALTPLL Megafunction

Unzip Design Example 2 (an507_altpll_dynphase_de2.zip) and compile it in the Quartus II software. Run a timing simulation using the **.vwf** file provided with the example. The example is configured to step up the phase by 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.

Conclusion

PLL reconfiguration is a powerful feature that you can use to vary the PLL clock output frequency and to shift the phase 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. The flexibility offered by the Cyclone III PLL makes it a superior clock management system.

Referenced Documents

This application note references the following documents:

- *Clock Networks and PLLs in Cyclone III Devices* chapter in volume 1 of the *Cyclone III 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*

Document Revision History

Table 5 lists the revision history for this document.

Table 5. Document Revision History

Date	Version	Changes
September 2011	2.0	■ 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 1
January 2008	1.0	Initial release.