

Introduction

Phase-locked loops (PLLs) use several divide counters and different voltage-controlled oscillator (VCO) phase taps to perform frequency synthesis and phase shifts. In Stratix® II enhanced and fast PLLs, you can reconfigure both the counter settings and phase-shift the PLL output clock in real time. You can also change the charge pump and loop filter components, which dynamically affects the PLL bandwidth. You can use these PLL components to update the output clock frequency, PLL bandwidth, and phase-shift in real time, without reconfiguring the entire FPGA.

The ability to reconfigure the PLL in real time is useful in applications that might operate at multiple frequencies. It is also useful in prototyping environments, allowing you to sweep PLL output frequencies and adjust the output clock phase on the fly. 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 the new PLL settings.

This application note discusses:

- The implementation of real-time PLL reconfiguration in Stratix II PLLs and steps through the software implementation of the PLL reconfiguration feature
- Design considerations that system designers must consider when selecting PLL parameters
- Design examples to help users better understand this feature

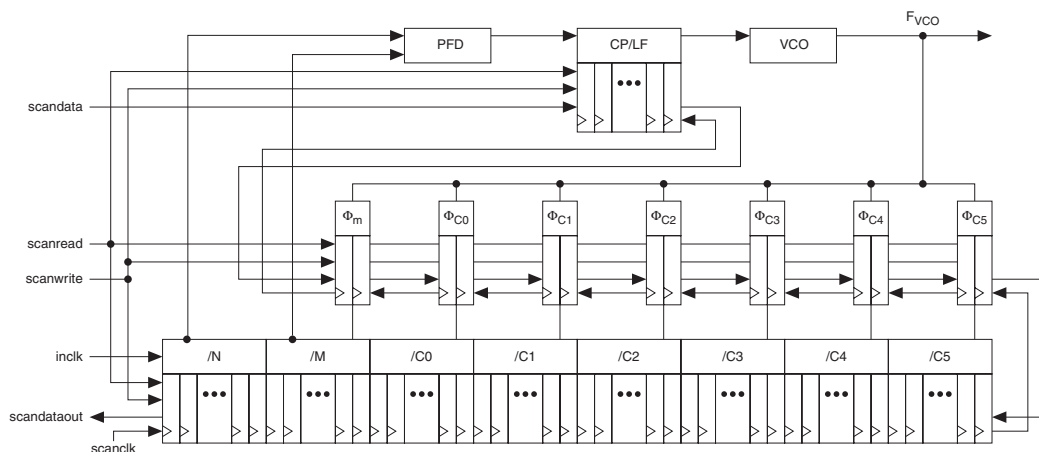
PLL Reconfiguration Hardware Implementation

Enhanced and fast PLLs in Stratix II devices support real-time PLL reconfiguration. The following PLL components are configurable in real time:

- Pre-scale counter (n)
- Feedback counter and VCO phase tap selection (m, ϕ_m)
- Post-scale output counters and VCO phase tap selection ($C0$ to $C5$, ϕ_{C0} to ϕ_{C5})
- Dynamically adjust the charge pump current (I_{cp}), loop filter components (R, C) to facilitate on the fly reconfiguration of the PLL bandwidth.

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 input to the scan chain via the `scandata` port and shift registers are clocked by `scanclk`. The maximum `scanclk` frequency is 100 MHz. After the last bit of data is clocked, asserting the `scanwrite` signal for at least one `scanclk` clock cycle causes the PLL configuration bits to be synchronously updated with the data in the scan registers.

Figure 1. PLL Reconfiguration Scan Chain *Note (1)*



Note to Figure 1:

- (1) The Stratix II fast PLLs do not support the C4 and C5 counters and the phase shift settings ϕ_{C4} and ϕ_{C5} .



The counter and phase shift settings are updated synchronous to the clock frequency of the individual counters. Therefore, all counters are not updated simultaneously.

Table 1 shows how these signals can be driven by the programmable logic device (PLD) logic array or I/O pins.

Table 1. Real-Time PLL Reconfiguration Ports			
PLL Port Name	Description	Source	Destination
scandata	Serial input data stream to scan chain.	Logic array or I/O pins	PLL reconfiguration circuit
scanclk	Serial clock input signal. This clock can be free running.	Logic array or I/O pins	PLL reconfiguration circuit
scanwrite	Writes the data in the scan chain to the PLL. Active high.	Logic array or I/O pins	PLL reconfiguration circuit
scanread	Enables scandata to be written into the scan chain. Active high.	Logic array or I/O pins	PLL reconfiguration circuit
scandone	Indicates when the PLL has finished reprogramming. A rising edge indicates the PLL has finished reprogramming.	PLL reconfiguration circuit	Logic array or I/O pins
scandataout	Used to output the contents of the scan chain.	PLL reconfiguration circuit	Logic array or I/O pins

All enhanced PLL counters except the m counter have 20 configuration bits. The m counter has 22 configuration bits. See Table 2 on page 5.

All fast PLL counters except the n counter have 12 configuration bits. The n counter has 3 configuration bits. See Table 3 on page 7.

There are two classes of counters, spread spectrum counters, and post-scale counters. The following two subsections describe these counters.

Spread Spectrum Counters (m , n)

The enhanced PLL pre-scale counter, n , and feedback counter, m , implement spread spectrum by switching between two different divide settings. These counters range from 1 to 511. Therefore, the nominal count value and the spread spectrum count value each need 9 configuration bits, for a total of 18 configuration bits. Two additional configuration bits are used to bypass the nominal and spread counters (for example, divide by 1). These two bypass bits must be set to the same value for proper operation. This brings the total number of counter configuration bits to 20. When spread spectrum is not used, the device uses the nominal count value and the spread spectrum count value is ignored. The m counter uses an additional two configuration bits to implement phase shift.



The Quartus® II software sets bits [9 . . 0] for the *m* and *n* spread counter setting to 0000000000 if the spread spectrum feature is not used. If this feature is used, bits [8 . . 0] show the corresponding spread count value. The spread count bypass setting, bit [9], is always set to 0.

Post-Scale Counters (C0 to C5)

The enhanced PLL, the C0 to C5 post-scale counters implement programmable duty cycle and do not implement the spread spectrum feature. See [Figure 3 on page 6](#). For enhanced PLLs, each counter has an 8-bit high time setting and an 8-bit low time setting. The duty cycle is the ratio of output high or low time to the total cycle time, which is the sum of the two. Additionally, these counters have two control bits, *rbypass*, for bypassing the counter, and *RSELODD*, to select the output clock duty cycle, and two control bits for up/down phase shift selection, bringing the total number of configuration bits to 20.

When the *rbypass* bit is set to 1, it bypasses the counter, resulting in a divide by 1. When this bit is set to 0, the high and low time counters are added to compute the effective division of the VCO output frequency. For example, if the post-scale divide factor is 10, the high and low count values could be set to 5 and 5 respectively, to achieve a 50-50% duty cycle. The PLL implements this duty cycle by transitioning the output clock from high to low on the rising edge of the VCO output clock. However, a 4 and a 6 setting for the high and low count values, respectively, would produce an output clock with 40-60% duty cycle.

The *rselodd* bit indicates an odd divide factor for the VCO output frequency along with a 50% duty cycle. For example, if the post-scale divide factor was 3, the high and low time count values could be set to 2 and 1 respectively to achieve this division. This implies a 67%-33% duty cycle. If you need a 50%-50% duty cycle, you can set the *RSELODD* control bit to 1 to achieve this duty cycle despite an odd division factor. The PLL implements this duty cycle by transitioning the output clock from high to low on a falling edge of the VCO output clock. When you set *RSELODD* = 1, you subtract 0.5 cycles from the high time and you add 0.5 cycles to the low time. For example:

- High time count = 2 cycles
- Low time count = 1 cycle
- *RSELODD* = '1' effectively equals:
 - High time count = 1.5 cycles
 - Low time count = 1.5 cycles
 - Duty cycle = (1.5/3) % high time count and (1.5/3) % low time count

Scan Chain Description

The length of the scan chain varies for different PLLs. Enhanced PLLs 5, 6, 11, and 12 have a 174-bit scan chain. Table 2 shows the number of bits for each component of the enhanced PLL. Figure 2 shows the scan chain order of PLL components for enhanced PLLs. Fast PLLs only have 6 counters and a 75-bit scan chain. Figure 5 on page 8 shows the scan chain order of each component for fast PLLs.

As Table 2 shows, scan registers for the counter settings are marked N, M, and C0 to C5. As Figure 1 on page 2 shows, scan registers for the phase shift are marked ϕ_m , ϕ_{C0} to ϕ_{C5} .

Table 2. Enhanced PLL Reprogramming Bits				
Block Name	Number of Bits			
	Counter	Phase	Other (1)	Total
C0	16	2	2	20
C1	16	2	2	20
C2	16	2	2	20
C3	16	2	2	20
C4	16	2	2	20
C5	16	2	2	20
M	18	2	2	22
N	18	0	2	20
Charge Pump	0	0	4	4
Loop Filter Resistor	0	0	6	6
Loop Filter Capacitor	0	0	2	2
Total number of bits				174

Note to Table 2:

- (1) Includes two control bits, *rbypass*, for bypassing the counter, and *rseledd*, to select the output clock duty cycle.

Figure 2 shows the scan chain order of PLL components for enhanced PLLs 5, 6, 11, and 12.

Figure 2. Scan Chain Order for Enhanced PLLs 5, 6, 11 & 12

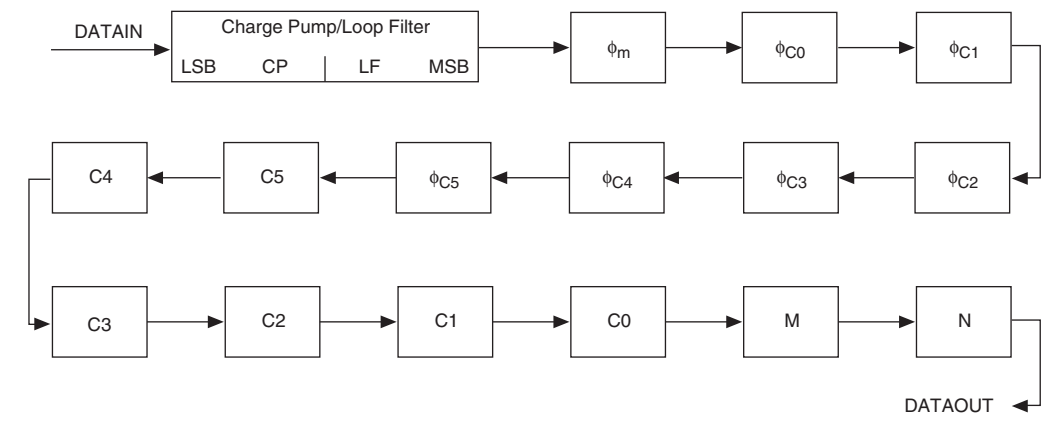
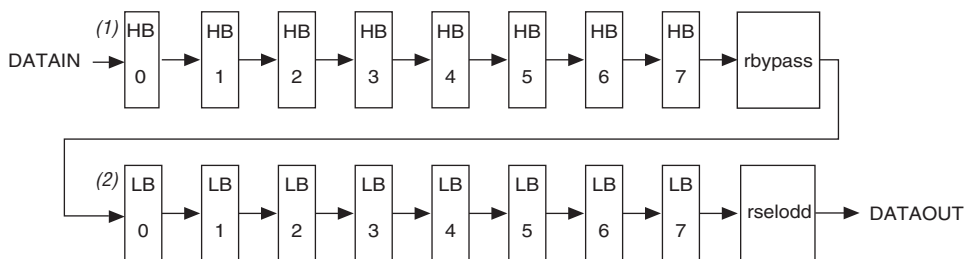


Figure 3 shows the scan chain bit order sequence for enhanced PLL post-scale counters.

Figure 3. Enhanced PLL Post-Scale Counter Bit Order

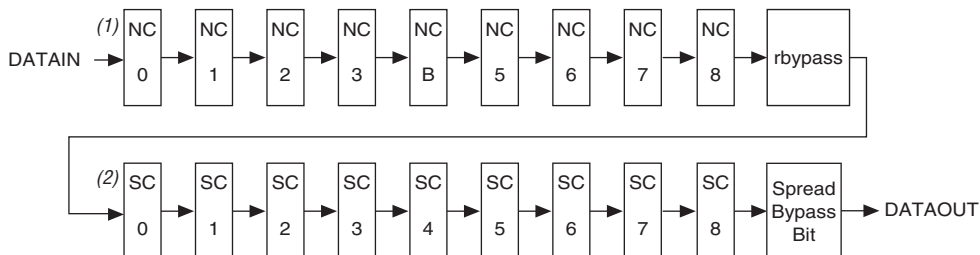


Notes to Figure 3:

- (1) HB: high bit.
- (2) LB: low bit.

Figure 4 shows the scan chain bit order sequence for the enhanced PLL spread-spectrum counters.

Figure 4. Spread Spectrum Counter Bit Order



Notes to Figure 4:

- (1) NC: nominal count.
- (2) SC: spread count.

Table 3 shows the number of bits for each reconfigurable component of the fast PLL.

Table 3. Fast PLL Reprogramming Bits				
Block Name	Number of Bits			
	Counter	Phase	Other (1)	Total
C0	8	2	2	12
C1	8	2	2	12
C2	8	2	2	12
C3	8	2	2	12
M	8	2	2	12
N	2	0	1	3
Charge Pump	0	0	4	4
Loop Filter Resistor	0	0	6	6
Loop Filter Capacitor	0	0	2	2
Total number of bits				75

Note to Table 3:

- (1) Includes two control bits, rbypass, for bypassing the counter, and rselodd, to select the output clock duty cycle.

Figure 5 shows the scan chain order of components in fast PLLs.

Figure 5. Scan Chain Order for Fast PLLs 1, 2, 3, 4, 7, 8, 9 & 10

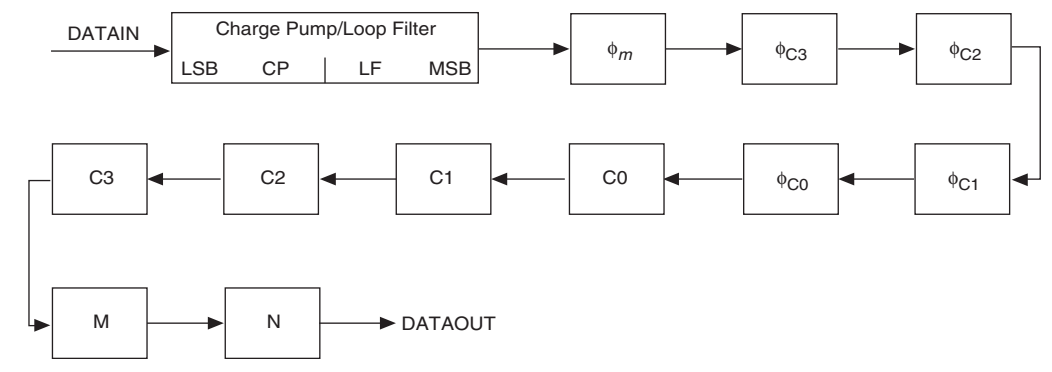
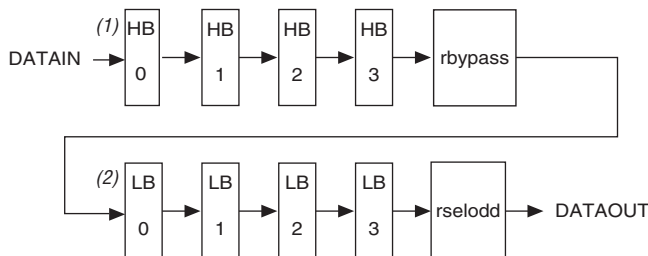


Figure 6 shows the scan chain bit order sequence for fast PLL post-scale, m counters.

Figure 6. Fast PLL Post-Scale Counter and m Counter Bit Order



Notes to Figure 6:

- (1) HB: high bit.
- (2) LB: low bit.

Reconfigurable Phase Shift

The Quartus II software automatically sets the phase taps and counter settings according to the phase-shift entry. By entering your desired phase-shift parameters, Quartus II software automatically sets the closest setting achievable. The fine phase shift is also reconfigurable during system operation. You can phase shift one, some, or all clock outputs in one operation. You can select phase-shifting values in time units with a resolution of 125 to 250 ps. This resolution is a function of frequency input

and the multiplication and division factors (for example, it is a function of the VCO period), with the smallest incremental step equal to an eighth (0.125) of the VCO period.

Each clock output counter can choose a different phase of the VCO period from up to eight taps for individual fine-step selection. In addition, each clock output counter can use a unique initial count setting to achieve individual coarse-shift selection in steps of one VCO period. The combination of coarse and fine grain shifts allows phase shifting for the entire output clock period.



The phase shift stepping feature allows users to dynamically select a different VCO phase tap, that is, the fine phase shift can be reconfigured. The coarse phase shift cannot be reconfigured.

Use the following equation to determine the precision of the phase shifting in degrees:

$$(45^\circ \text{ of the } V_{CO}) \div \text{Post-scale counter value} = \text{precision of the phase shifting in degrees}$$

Therefore, the maximum step size is 45° of the V_{CO} . Smaller steps are possible depending on the division ratio on the output counter port. Implementing phase shift as described above provides the highest precision, since it is the least sensitive to process, voltage, and temperature variation.

Figure 7 shows how the phase shift stepping feature works in Stratix II devices. For example, consider the case where the counter is using the 0° phase tap as the reference source. The user programs the counter to take one phase step forward. On the falling edge of the 0° phase, the counter reference clock switches from the 0° phase tap to the 45° phase tap so the next rising edge the counter sees is the 45° phase tap.

Figure 7. PLL Phase Shifting Implementation, Example 1

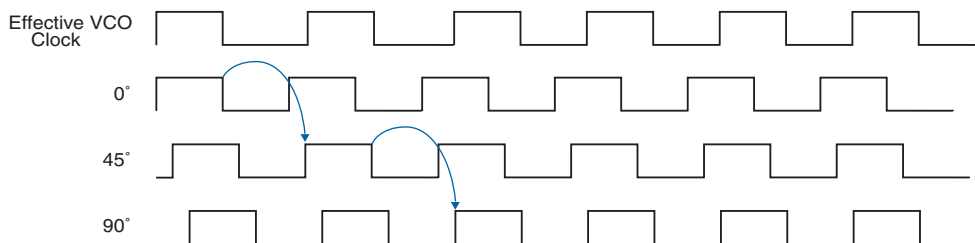
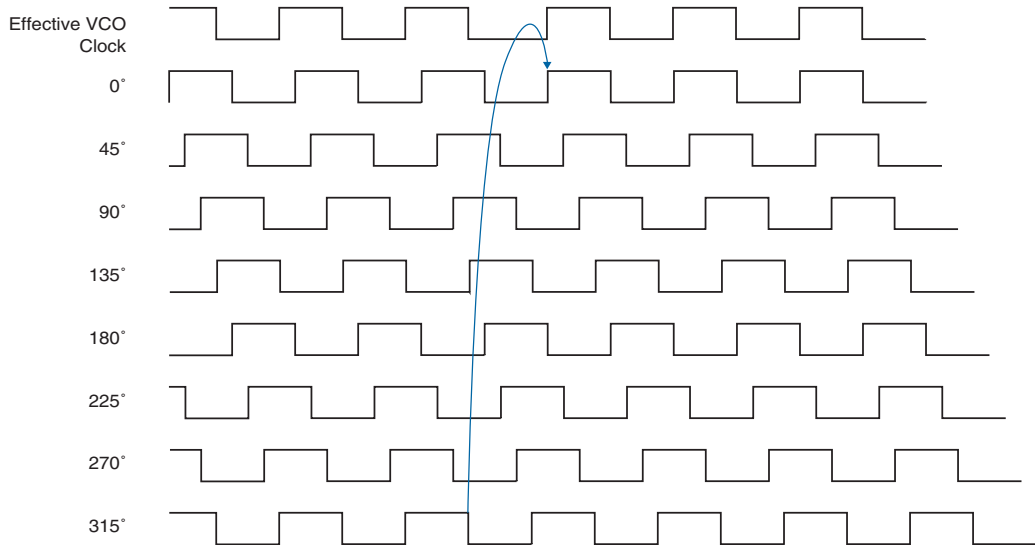


Figure 8 shows how the phase shift is continuous since stepping from the 315° phase tap to the 0° phase tap continues shifting the phase in the same direction.

Figure 8. PLL Phase Shifting Implementation, Example 2



You can take the same approach to shift the phase in the opposite direction, as shown by the arrows in Figure 9.

Figure 9. PLL Phase Shifting Implementation Reverse, Example 3

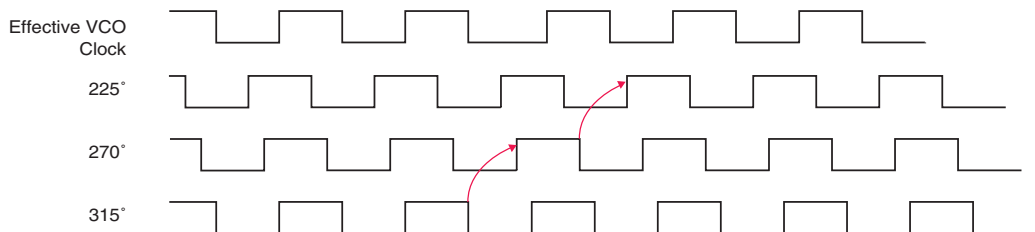
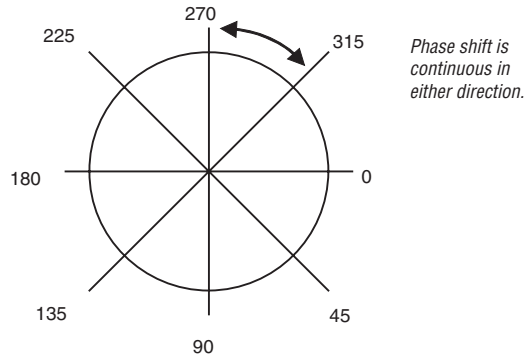


Figure 10 shows the continuous phase shift from another perspective. The phase shift is continuously shifting in either the clockwise or counter clockwise direction.

Figure 10. Reconfigurable Phase Shift



When programming the device, an initial phase tap of the VCO is selected for each counter (m , C0 to C5). You can then change the phase tap (either forward or reverse) using the PLL reconfiguration interface. If the PLL is reset by toggling the `areset` signal or reconfiguring the FPGA, the phase is set back to the original phase shift in the configuration file.

Table 4 shows the phase control bit definitions for Stratix II PLLs.

Table 4. Stratix II PLL Phase Control Bit Definitions		
$\Phi B1$ (Direction)	$\Phi B0$ (Step Enable)	Action Taken
X	0	No phase shift stepping. Reconfigurable (fine) phase shift is disabled.
0	1	Step back one phase step
1	1	Step forward one phase step

Charge Pump & Loop Filter

You can reconfigure the charge pump and loop filter settings to update the PLL bandwidth on the fly. [Tables 5, 6, and 7](#) show the charge pump and loop filter values you can set for the Stratix II Enhanced PLLs.

Table 5. Charge Pump Bit Control

Control Bits				Pumping Current Icp (μ A)
CP3	CP2	CP1	CP0	
0	0	0	1	12
0	0	1	0	30
0	0	1	1	36
0	1	0	0	52
0	1	0	1	57
0	1	1	0	72
0	1	1	1	77
1	0	0	0	92
1	0	0	1	96
1	0	1	0	110
1	0	1	1	114
1	1	0	0	127
1	1	0	1	131
1	1	1	0	144
1	1	1	1	148

Table 6. Loop Filter Resistor Value Control (Part 1 of 2)

Control Bits						Resistance (Ω)
LF5	LF4	LF3	LF2	LF1	LF0	
0	0	0	0	0	0	1,000
0	0	0	0	0	1	1,500
0	0	0	0	1	0	2,000
0	0	0	0	1	1	2,500
0	0	0	1	0	0	3,000
0	0	0	1	0	1	3,500
0	0	0	1	1	0	4,000
0	0	0	1	1	1	4,500
0	0	1	0	0	0	5,000

Table 6. Loop Filter Resistor Value Control (Part 2 of 2)

Control Bits						Resistance (Ω)
LF5	LF4	LF3	LF2	LF1	LF0	
0	0	1	0	0	1	5,500
0	0	1	0	1	0	6,000
0	0	1	0	1	1	6,500
0	0	1	1	1	1	8,500
0	1	1	0	0	0	9,000
0	1	1	0	0	1	9,500
0	1	1	0	1	0	10,000
0	1	1	0	1	1	10,500
0	1	1	1	0	0	11,000
0	1	1	1	0	1	11,500
0	1	1	1	1	0	12,000
0	1	1	1	1	1	12,500
1	0	1	0	0	0	13,000
1	0	1	0	0	1	13,500
1	0	1	0	1	0	14,000
1	0	1	0	1	1	14,500
1	0	1	1	0	0	15,000
1	0	1	1	0	1	15,500
1	0	1	1	1	0	16,000
1	0	1	1	1	1	16,500
1	1	1	0	0	0	17,000

Table 7. Loop Filter Control of the High Frequency Capacitor

Control Bits		Capacitance (pF)
LF7	LF6	
0	1	16
1	1	5

Tables 8, 9, and 10 show the charge pump and loop filter values you can set for the Stratix II Fast PLLs.

Table 8. Charge Pump Bit Control				
Control Bits				Pumping Current I_{cp} (μA)
CP3	CP2	CP1	CP0	
0	0	0	1	12
0	0	1	0	30
0	0	1	1	36
0	1	0	0	52
0	1	0	1	57
0	1	1	0	72
0	1	1	1	77
1	0	0	0	92
1	0	0	1	96
1	0	1	0	110
1	0	1	1	114
1	1	0	0	127
1	1	0	1	131
1	1	1	0	144
1	1	1	1	148

Table 9. Loop Filter Resistor Value Control (Part 1 of 2)						
Control Bits						Resistance Ω
LF5	LF4	LF3	LF2	LF1	LF0	
0	0	0	0	0	0	1,000
0	0	0	0	0	1	1,500
0	0	0	0	1	0	2,000
0	0	0	0	1	1	2,500
0	0	0	1	0	0	3,000
0	0	0	1	0	1	3,500

Table 9. Loop Filter Resistor Value Control (Part 2 of 2)

Control Bits						Resistance Ω
LF5	LF4	LF3	LF2	LF1	LF0	
0	0	0	1	1	0	4,000
0	0	0	1	1	1	4,500
0	0	1	0	0	0	5,000
0	0	1	0	0	1	5,500
0	0	1	0	1	0	6,000
0	0	1	0	1	1	6,500
0	0	1	1	0	0	7,000
0	0	1	1	0	1	7,500
0	0	1	1	1	0	8,000
0	0	1	1	1	1	8,500
0	1	1	0	0	0	9,000
0	1	1	0	0	1	9,500
0	1	1	0	1	0	10,000

Table 10. Loop Filter Control of the High Frequency Capacitor

Control Bits		Capacitance (pF)
LF7	LF6	
1	0	8
1	1	2

Bypassing PLL Counters

Bypassing a PLL counter results in a multiply (m counter) or a divide (n , C0 to C5 counters) factor of one.

The only way to bypass an m or n counter is to set the bypass bit for that counter to 1 and set the counter's least significant bit (LSB) to zero.

Table 11 shows the m and n counter settings for enhanced PLLs.

Table 11. <i>m</i> , <i>n</i> Counter Settings										
Enhanced PLL Scan Chain Bits [0..9] Settings										Description
LSB (2)									MSB (1)	
0	X	X	X	X	X	X	X	X	1 (3)	PLL counter bypassed
1	X	X	X	X	X	X	X	X	1 (3)	PLL counter disabled
x	X	X	X	X	X	X	X	X	0 (3)	PLL counter not bypassed or disabled because [9] (MSB) is set to 0

Notes to Table 11:

- (1) MSB: most significant bit.
- (2) LSB: least significant bit.
- (3) Bypass bit.

For only the m and n counters, if you set the bypass bit high and set the count value to 1 (or any number that ends with 1), it disables the counter and renders the PLL inoperable.

To bypass any of the 6 output counters (C0 to C5 counters), set the bypass bit to 1. The values on the other bits will be ignored.



To set m and n counters to 1, the Quartus II software always selects the PLL bit setting "000000001" as shown in the first row of Table 11.

Implementing Reconfigurable PLLs in the Quartus II Software

You can specify the PLL I/O frequencies and phase shifts in the Quartus II software `altpll` MegaWizard® Plug-In Manager. Based on these parameters, the Quartus II software picks internal settings for the PLL. These internal settings will be stored in the configuration file and used by the PLL after power-up and configuration.



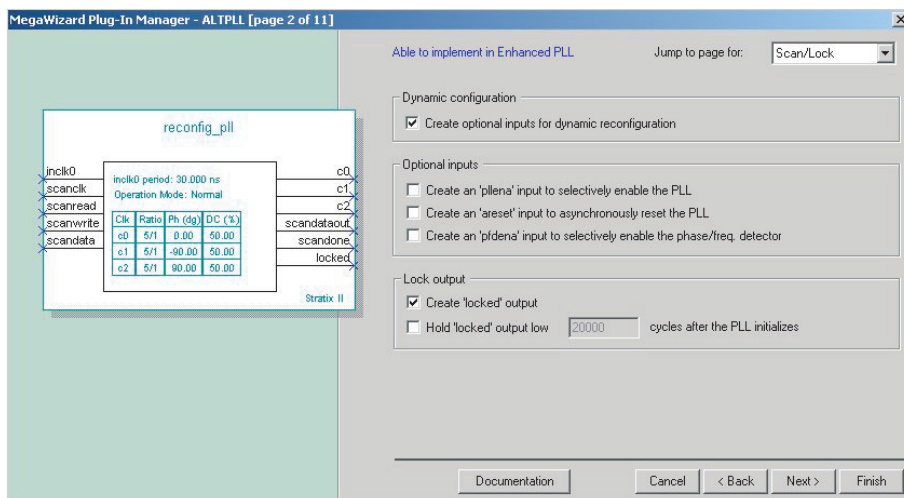
Starting in Quartus II software version 5.0, the `PRESERVE_PLL_COUNTER_ORDER` setting in the Quartus II software logic options is automatically turned on if the PLL reconfiguration feature is used in the PLL.

Because this option is on, Quartus II software will not rotate counters to improve clock routing, and there may be some clock routing errors as a result. To work around them, users can first turn off PLL reconfiguration and let Quartus II software pick the optimal counter selection. Then turn PLL reconfiguration back on and change the PLL outputs to use the counter selection as chosen by the Quartus software (by moving `c0` to `c1`, etc.).

altpll MegaWizard Plug-In Manager

You can use the `altpll` MegaWizard Plug-In Manager to enable the PLL reconfiguration circuitry, as shown in Figure 11. Enabling the reconfiguration circuitry automatically adds the `scanclk`, `scanread`, `scanwrite`, `scandata`, `scandataout`, and `scandone` ports to the `altpll` instance.

Figure 11. Enabling PLL Reconfiguration in the Quartus II MegaWizard Plug-In Manager



altpll_reconfig Megafunction

The `altpll_reconfig` megafunction simplifies the process to reconfigure Stratix II enhanced or fast PLLs on the fly. You can use the `altpll_reconfig` MegaWizard Plug-In Manager to reconfigure Stratix II enhanced or fast PLLs.

Figures 12 and 13 show the MegaWizard interface for the `altpll_reconfig` megafunction. You can access this megafunction in the Quartus II software through `libraries\megafunctions\IO\altpll_reconfig`.

Figure 12. MegaWizard Plug-In Manager Interface 1

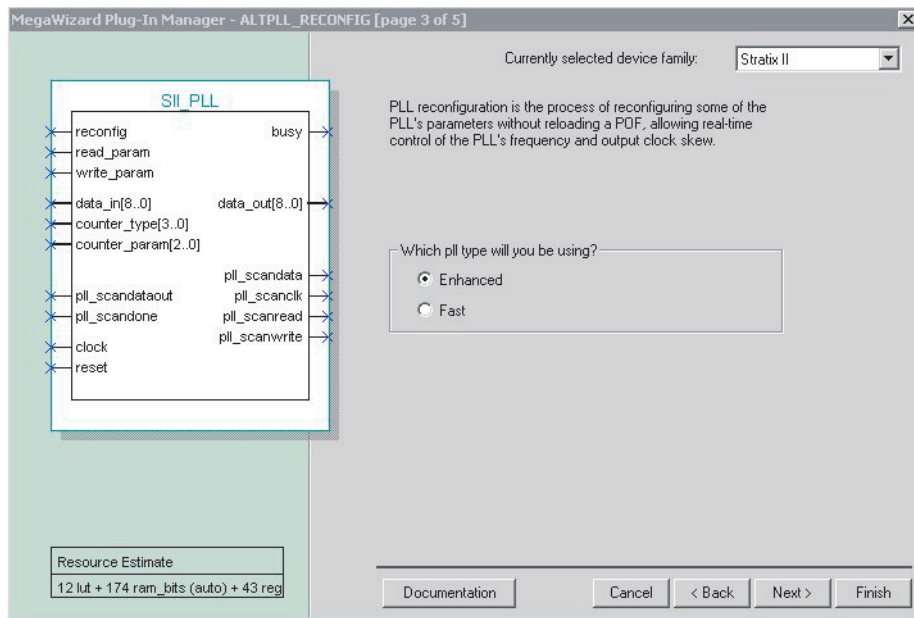
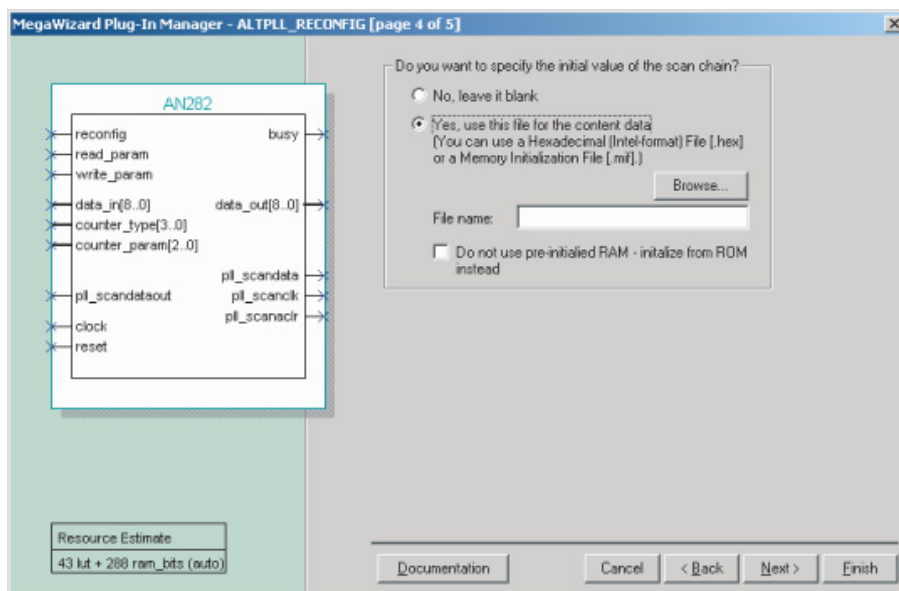


Figure 13. MegaWizard Plug-In Manager Interface 2

You must specify the type of the PLL: either an Enhanced PLL (174 bits for PLL's 5, 6, 11, and 12) or a Fast PLL (75 bits for PLL's 1, 2, 3, 4, 7, 8, 9, and 10). You can also specify a memory initialization file (.mif) commonly called MIF, or a hexadecimal (.hex) file with the new PLL settings and transfer it into the PLL scan registers. Alternatively, you can choose to leave the initial value of the scan chain as blank, which means that the device powers up with all the PLL counters being programmed with zero's.



Hardcopy II devices do not support pre-initialized RAM so you can choose to initialize from ROM instead. The check box for the **Do not use pre-initialized RAM - initialize from ROM instead** option is specifically intended for Hardcopy II devices. When the Hardcopy II device powers up, the contents of the ROM are copied to a RAM which is then used to load the PLL counters.

MIF Generation

To increase the usability of the `altpll_reconfig` megafunction feature, the Quartus II software compiler generates a MIF that references the `altpll` parameter settings when PLL reconfiguration is on in the `altpll` MegaWizard Plug-In Manager. You can now use the MIF to initialize the scan chain (for example, the `INIT_MIF_FILE` parameter) of the `altpll_reconfig` megafunction.

To reconfigure the PLL, you must connect the `scanclk` port on the `altpll` instance. This enables the feature on the `altpll` instance. The Quartus II software only generates a MIF when reconfiguring the PLL. The PLL instance name derives the name of the generated MIF. This name is in the Compilation Report, under the PLL Summary Section.

Use the following steps to implement the `altpll_reconfig` megafunction.

1. Create an `altpll` instance with your choice of multiplication/division factors and turn **ON** the PLL reconfiguration feature.
2. Create an `altpll_reconfig` instance and specify the PLL type (enhanced or fast) to match the `altpll` instance created in Step 1.
3. Compile (or synthesize) the design in the Quartus II software to generate a MIF representing the initial or default state of the PLL scan chain. Optionally, change the MIF to account for the counters/phase shift settings that will be change during PLL reconfiguration.
4. Edit the `altpll_reconfig` function instance and specify the `init_scan_file` parameter to the MIF that you modified in Step 3.
5. Connect the output ports from the `altpll_reconfig` instance to the input ports of the `altpll` instance as shown in Figure 14. Not all connections are required. See the “Ports & Parameters” section for more information on the `altpll_reconfig` I/O ports.



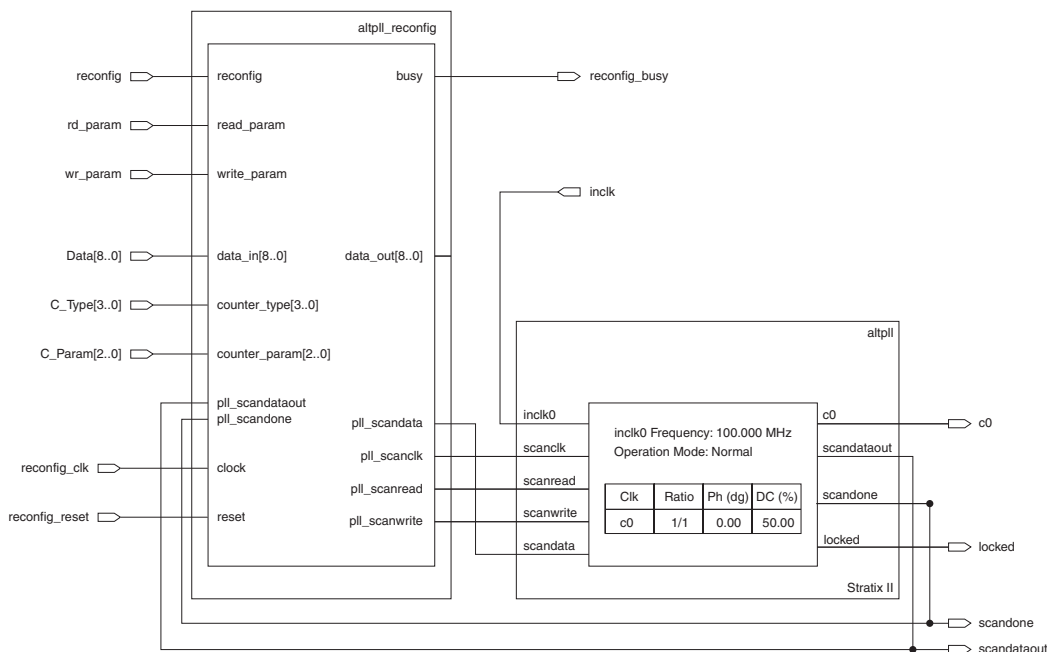
The MIF file generated by Quartus II software is intended for post layout simulation only. Using this MIF file to perform reconfiguration simulation using the behavioral model of the PLL could result in erroneous behavior as the counter settings determined by the `altpll` behavioral model may not match those determined by Quartus II software after a successful compilation.



See the *altpll Megafunction User Guide* for additional information on the altpll I/O ports.

Figure 14 shows that the pll_scanclk, pll_scanread, pll_scanwrite, and pll_scandata outputs from the altpll_reconfig megafunction feed the input ports used for PLL reconfiguration (for example, scanclk, scanread, scanwrite, and scandata).

Figure 14. Connecting the altpll_reconfig Block to the altpll Block



The `altpll_reconfig` megafunction simplifies the process of reconfiguring the PLL. The `altpll_reconfig` megafunction has a state machine that takes care of operations like asserting the busy signal during read/write operations into the scan chain, asserting `scanwrite` after clocking the last bit into the scan chain.



Users should be careful when modifying the MIF for reconfiguring the counter settings. An invalid setting (bypass bit = 1 and LSB of the n, m counter = 1) could cause the PLL counters to be disabled after reconfiguration. This is shown in the second row of Table 11 on page 16.

PLL Configuration Scan Register Bit Map

Advanced PLL users can manually select the counter and phase shift settings based on information detailed in “[PLL Reconfiguration Hardware Implementation](#)” on page 2. After determining the individual configuration bit settings for the different counters and phase shift, arrange the bits as shown in [Table 12](#).

[Table 12](#) provides a bit map for the enhanced PLL scan chain registers. The last bit shifted into the scan chain is Bit 0. Bit 173 is the first bit shifted in for enhanced PLLs 5, 6, 11, and 12. Bit 74 is the first bit to be shifted in for fast PLLs 1, 2, 3, 4, 7, 8, 9, and 10.

Table 12. Enhanced PLL Configuration Scan Chain Bit Map (Part 1 of 2)

PLL Scan Chain Bit Map (1)									PLL Component	Size (Bits)
LSB				MSB						
					0	1	2	3	Charge pump settings [3:0]	4
	4	5	6	7	8	9	10	11	Loop filter settings [11:4]	8
							12	13	m phase shift setting	2
							14	15	C0 counter phase shift setting	2
							16	17	C1 counter phase shift setting	2
							18	19	C2 counter phase shift setting	2
							20	21	C3 counter phase shift setting	2
							22	23	C4 counter phase shift setting	2
							24	25	C5 counter phase shift setting	2
	26	27	28	29	30	31	32	33	C5 counter high cycles count	8
								34	C5 counter bypass bit	1
	35	36	37	38	39	40	41	42	C5 counter low cycles count	8
								43	C5 counter odd division bit	1
	44	45	46	47	48	49	50	51	C4 counter high cycles count	8
								52	C4 counter bypass bit	1
	53	54	55	56	57	58	59	60	C4 counter low cycles count	8
								61	C4 counter odd division bit	1
	62	63	64	65	66	67	68	69	C3 counter high cycles count	8
								70	C3 counter bypass bit	1
	71	72	73	74	75	76	77	78	C3 counter low cycles count	8
								79	C3 counter odd division bit	1
	80	81	82	83	84	85	86	87	C2 counter high cycles count	8
								88	C2 counter bypass bit	1
	89	90	91	92	93	94	95	96	C2 counter low cycles count	8

Table 12. Enhanced PLL Configuration Scan Chain Bit Map (Part 2 of 2)

PLL Scan Chain Bit Map (1)									PLL Component	Size (Bits)
LSB								MSB		
								97	C2 counter odd division bit	1
	98	99	100	101	102	103	104	105	C1 counter high cycles count	8
								106	C1 counter bypass bit	1
	107	108	109	110	111	112	113	114	C1 counter low cycles count	8
								115	C1 counter odd division bit	1
	116	117	118	119	120	121	122	123	C0 counter high cycles count	8
								124	C0 counter bypass bit	1
	125	126	127	128	129	130	131	132	C0 counter low cycles count	8
								133	C0 counter odd division bit	1
134	135	136	137	138	139	140	141	142	<i>m</i> counter nominal count	9
								143	<i>m</i> counter bypass bit	1
144	145	146	147	148	149	150	151	152	<i>m</i> counter spread count	9
								153	<i>m</i> spread counter bypass bit	1
154	155	156	157	158	159	160	161	162	<i>n</i> counter nominal count	9
								163	<i>n</i> counter bypass bit	1
164	165	166	167	168	169	170	171	172	<i>n</i> counter spread count	9
								173	<i>n</i> spread counter bypass bit	1
									Scan chain length	174

Note to Table 12:

- (1) All registers in the scan chain are setup such that the most signification bit (MSB) is shifted in first and the LSB last.

Table 13 provides a bit map for the fast PLL scan chain registers.

Table 13. Fast PLL Configuration Scan Chain Bit Map (Part 1 of 2)

PLL Scan Chain Bit Map (1)									PLL Component	Size (Bits)
LSB								MSB		
				0	1	2	3		Charge pump settings [3:0]	4
4	5	6	7	8	9	10	11		Loop filter settings [11:4]	8
						12	13		<i>m</i> phase shift setting	2
						14	15		C3 counter phase shift setting	2
						16	17		C2 counter phase shift setting	2
						18	19		C1 counter phase shift setting	2

Table 13. Fast PLL Configuration Scan Chain Bit Map (Part 2 of 2)

PLL Scan Chain Bit Map (1)								PLL Component	Size (Bits)
LSB				MSB					
						20	21	C0 counter phase shift setting	2
				22	23	24	25	C0 counter high cycles count	4
							26	C0 counter bypass bit	1
				27	28	29	30	C0 counter low cycles count	4
							31	C0 counter odd division bit	1
				32	33	34	35	C1 counter high cycles count	4
							36	C1 counter bypass bit	1
				37	38	39	40	C1 counter low cycles count	4
							41	C1 counter odd division bit	1
				42	43	44	45	C2 counter high cycles count	4
							46	C2 counter Bypass bit	1
				47	48	49	50	C2 counter low cycles count	4
							51	C2 counter odd division bit	1
				52	53	54	55	C3 counter high cycles count	4
							56	C3 counter bypass bit	1
				57	58	59	60	C3 counter low cycles count	4
							61	C3 counter odd division bit	1
				62	63	64	65	m counter high cycles count	4
							66	m counter bypass bit	1
				67	68	69	70	m counter low cycles count	4
							71	m counter odd division bit	1
						72	73	n counter nominal count	2
							74	n counter bypass bit	1
								Scan chain length	75

Note to Table 13:

(1) All registers in the scan chain are setup such that the MSB is shifted in first and the LSB last.

Design Considerations

You must consider the following information when using PLL reconfiguration.

- Changing pre-scale and feedback counter settings (m, n) will affect the PLL VCO frequency, which may require the PLL to relock to the reference clock. Changing the ϕ_m phase shift setting will change 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 resynchronization.
- If you choose not to specify a MIF to specify the initial contents of the scan chain, then the scan chain is blank when the device enters user mode. Therefore you will need to update all the PLL counter parameters into the scan chain before pulsing the reconfig signal. Reconfiguring the PLL counters or phase shift while the scan chain is empty could cause reconfiguration to fail and the PLL to lose lock.
- Adding phase shift using the ϕ_m phase shift setting would pull in all the PLL clock outputs with respect to reference clock, effectively adding a negative phase shift. This is because (ϕ_m) is in the feedback path.
- When making changes to the loop elements (m, n, ϕ_m), Altera® recommends disabling PLL outputs to the logic array using the `clkena` signals available on the `altclkctrl` megafunction. This will eliminate the possibility of an over-frequency condition affecting system logic.
- Changes to post-scale counters and phase will not affect the PLL lock or VCO frequency. The resolution of phase shift is always a function of VCO frequency, with the smallest incremental step equal to an eighth (0.125) of the VCO period.
- When the phase relationship between output clocks is important, Altera recommends resynchronizing the PLL using the `areset` signal. This will reset all internal PLL counters and re-initiate the locking process. The PLL lock time depends on the loop bandwidth. For example, the higher the bandwidth, the faster the lock process. You can selectively change the loop parameters (Icp, R, C) to facilitate on the fly reconfiguration of the bandwidth.
- The Stratix II scan chain supports a free running `scanclk`, so there is no need to precisely control the start and stop of the clock. In addition, the `scanwrite` signal initiates PLL re-programming. Once the PLL is successfully reconfigured, toggling the `scanwrite` multiple times, phase shifts the output clock in increments of 45° or one eighth (0.125) of the VCO period.

- The `scanread` signal synchronously enables or disables the clock to the scan chain. The `scanread` signal must go high at least one `scanclk` cycle before you start reading `scandata` into the scan chain. Toggling `scanwrite` automatically disables the chain until reprogramming completes. The `scanread` signal can be left active all the time, but Altera recommends deasserting `scanread` when not using the reconfigure feature to reduce power consumption and to prevent inadvertent clocking of the scan chain.
- If the PLL loses lock during or after PLL reconfiguration, the m, n counter settings could have changed during the reconfiguration process. Additionally, if you changed the m, n counter/phase shift settings, the PLL could lose lock.

For example, if the input clock frequency is 350 MHz and output clock frequency is 700 MHz, the Quartus II software could choose from several ways to achieve these frequencies. For example, the Quartus II software could choose $m = 2, n = 1$. This results in a VCO frequency of 700 MHz and $C0 = 1$ to achieve the above frequency combination.

During reconfiguration, if you set your `scandata` bits to get $m = 4, n = 2$, you still keep the same input/output frequency combinations, however, this changed the m, n counter values and therefore, the PLL will lose lock. Instead of manually manipulating the scan bit values, Altera recommends that you allow the Quartus II software to perform the calculations.

See the Quartus II software PLL Summary section of the Compilation Report to see what values the Quartus II software chose for m, n to see if these settings were not mistakenly altered during PLL Reconfiguration.

- If you reconfigure one of the output counters to run, for example, at one fifth the original clock frequency, you only affect the other clock outputs if you also changed the m or n counter values. After reconfiguration, only the output counter that you reconfigured (without changing the m or n counter values) will reflect the new frequency.

Phase Shift Stepping

The `scanread` signal can be used to hold the value of the scan chain so it is not shifted out (gates the scan chain clock off). This allows writing the content of the scan chain into the PLL multiple times by toggling the `scanwrite` signal without re-writing the entire scan chain. This feature allows the user to quickly step the phase shift in a single direction by writing the appropriate scan data (for counter selection and phase shift direction) and then toggling the `scanwrite` signal the desired number of times to generate the desired phase shift. If the user wants to change the

phase shift direction, then you must reload the scan chain with the appropriate phase control bit setting as shown in [Table 4 on page 11](#). Toggling the `scanwrite` signal also automatically disables the scan chain until reprogramming is complete.

Use the following steps to reconfigure a PLL. [Figure 15](#) shows the outcome of this reconfiguration.

1. Disable all PLL outputs using the `clkena` signals available on the `altclkctrl` megafunction.



If you cannot disable the PLL outputs, you must make gradual and incremental changes to internal components. For example, if the reference clock input is 100 MHz and the counters n and m are set to 5 and 25 respectively, the VCO will run at 500 MHz. If you want to change the VCO frequency to 600 MHz, set the n and m counters to 5 and 30, respectively. You should gradually change the feedback counter (m) from 25 to 30 in increments of 1. This reduces the risk of an over frequency condition (where the output frequency is higher than required) and could avoid a loss of lock condition.

2. The `scanread` signal must go high at least one `scanclk` cycle before data is read into the chain.
3. Scan in the new counter and phase shift settings through the `scandata` port.
 - a. Transition the `scanwrite` signal high when the last bit is clocked into the scan chain (174th rising edge of `scanclk` for enhanced PLLs and 75th rising edge of `scanclk` for fast PLLs).
 - b. After the entire PLL reprogramming scan chain is loaded into the serial registers, they must still be synchronously shifted into the actual PLL registers. Thus, `scanread` must be disabled after the last data bit (174th bit for enhanced PLLs and 75th bit for fast PLLs) in the scan chain is clocked in prior to initiating the parallel transfer by the `scanwrite` signal to complete the reprogramming.
4. This step is optional. If you do not perform this step, go to [step 5](#). Assert the `scanwrite` signal the desired number of times to generate the desired phase shift. Re-enable the PLL output and do not reset the PLL. If you perform this step, this is the last step for reconfiguring the phase shift.

5. Reset the PLL using `areset` to maintain phase relationships between output clocks. Do not reset the PLL when using the phase shift stepping feature. Resetting the PLL will cause the phase shift to revert back to the value programmed into the SRAM Object File (SOF) or Programming Object File (POF).
6. Re-enable PLL outputs after detecting a valid lock.

After asserting the `scanwrite` signal, the `scandone` signal goes high after about 3 `scanclk` cycles. A falling edge on the `scanwrite` signal causes the `scandone` signal to go low.

Altera recommends waiting an additional `scanclk` cycle after `scandone` goes high to ensure that the new PLL settings are properly updated.

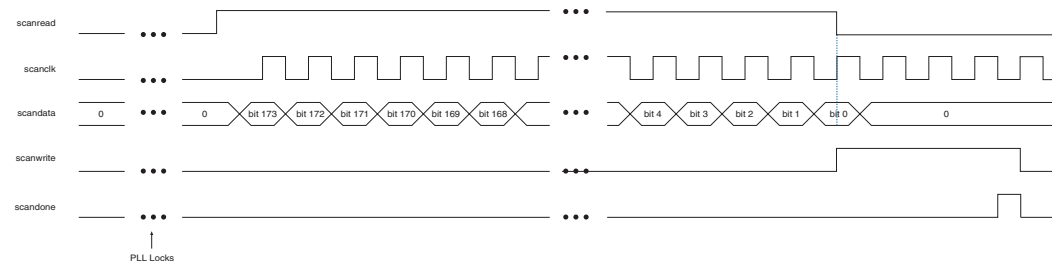
Please refer to the *Stratix II FPGA Family Errata Sheet* for more information on the `scandone` signal.



If you choose to use the `altpll_reconfig` megafunction, then all the `scandata` bits (174 for EPLLs and 75 for FPLLs) are shifted into the scan chain. Users that prefer to not use the `altpll_reconfig` megafunction can shift `scandata` a few bits at a time thereby reducing the size of the shift register.

Figure 15 shows a functional simulation of the PLL reconfiguration feature.

Figure 15. PLL Reconfiguration Waveform



Ports & Parameters

Tables 14 through 18 describe the parameters, input ports, output ports, and counter settings for the `altpll_reconfig` megafunction.

Table 14. `altpll_reconfig` Megafunction Parameters

Parameter Name	Description
<code>PLL_TYPE</code>	Defines the type of PLL that the megafunction will control. Legal values are either <code>enhanced</code> or <code>fast</code> . The enhanced PLLs have 174 bits of configuration, while fast PLLs have 75-bits of configuration.
<code>SCAN_INIT_FILE</code>	Name of the MIF or Intel-Format (HEX) file to be used as the initial value of the scan chain. This file can either be 174 bits or 75 bits depending on the <code>PLL_TYPE</code> . The format of the data bits must follow the format of the scan chain as specified in Table 12 or Table 13. If not specified, then the scan chain will be initialized with the default of 0.

Table 15. `altpll_reconfig` Megafunction Input Ports (Part 1 of 3)

Port Name	Required	Description
<code>clock</code>	Yes	Clock input used to load individual parameters, as well as to drive the PLL during reconfiguration. This port must be connected to a valid clock.
<code>data_in[]</code>	No	Nine-bit bus through which the parameter data can be provided when writing parameters. Some parameters do not use all nine bits. In this case, only the bits starting from bit 0 are used (for example, phase step setting will use bits 0-1). This bus defaults to 0 if left unconnected.
<code>counter_type[3..0]</code>	No	Four-bit bus that selects which counter type should be updated. Table 17 shows the mapping to determine which counter is specified for each <code>counter_type</code> value.
<code>counter_param[2..0]</code>	No	Three-bit bus that selects which parameter for the given counter type should be updated. Table 18 lists the mapping to each parameter type.

Table 15. *altpll_reconfig* Megafunction Input Ports (Part 2 of 3)

Port Name	Required	Description
<code>read_param</code>	No	<p>Signal indicating that the parameter specified with <code>counter_type[]</code> and <code>counter_param[]</code> should be read from the scan chain and fed to <code>data_out[]</code>. The number of bits read and set on <code>data_out[]</code> is dependant on the parameter type as indicated above. This signal is sampled at the rising clock edge, at which point the parameter value begins to be read from the scan chain. Additionally, this signal should only be asserted for one clock cycle to prevent the parameter from being reread on any subsequent clock cycle.</p> <p>The <code>busy</code> signal will be activated as soon as <code>read_param</code> is read and asserted. While the parameter is being read, the <code>busy</code> signal remains asserted. Once the <code>busy</code> signal is deactivated, <code>data_out[]</code> will be valid and the next parameter can begin to be loaded. While the <code>busy</code> signal is activated, <code>data_out[]</code> is not valid.</p>
<code>write_param</code>	No	<p>Signal indicating that the parameter specified with <code>counter_type[]</code> and <code>counter_param[]</code> should be written into the scan chain with the value given in <code>data_in[]</code>. The number of bits read from <code>data_in[]</code> is dependant on the parameter type as indicated above. This signal is sampled at the rising clock edge, at which point the parameter value begins to be written into the scan chain. Additionally, this signal should only be asserted for one clock cycle to prevent the parameter being rewritten on any subsequent clock cycle.</p> <p>The <code>busy</code> signal will be activated as soon as <code>write_param</code> is read and asserted. While the parameter is written, the <code>busy</code> signal remains asserted, and the input to <code>datain[]</code> will be ignored. Once the <code>busy</code> signal is deactivated, the device begins to write the next parameter.</p>

Table 15. *altpll_reconfig* Megafunction Input Ports (Part 3 of 3)

Port Name	Required	Description
<code>reconfig</code>	Yes	<p>Signal indicating that the PLL should be reconfigured with the new PLL settings as specified in the scan chain. The device samples this signal at the rising edge of the clock, at which point the scan chain settings begin to be loaded into the PLL. Additionally, you should only assert this signal for one clock cycle to prevent the PLL being reloaded after reconfiguration is complete.</p> <p>The <code>busy</code> signal will be activated as soon as <code>reconfig</code> signal is read as activated. The <code>pll_scanwrite</code> signal is also activated to start loading the new settings from the scan chain into the PLL. While the PLL is being reconfigured, the <code>busy</code> signal remains activated. Once the <code>busy</code> signal is deactivated, the parameters can be modified again.</p> <p>Altera recommends setting the counter and phase step control bits and then transition the <code>reconfig</code> signal high. This sends the <code>scandata</code> into the scan chain only once. See Figure 14 on page 21. Toggling the <code>reconfig</code> signal the desired number of times toggles the <code>scanwrite</code> signal, thereby incrementing or decrementing the output clock (<code>c0</code>) phase shift.</p>
<code>reset</code>	Yes	Asynchronous reset input used to initialize the machine such that it is in a valid state. The machine must be reset before first use otherwise the state is not guaranteed to be valid. This port must be connected.
<code>pll_scandataout</code>	No	Input signal to be driven by the <code>scandataout</code> port on the <code>altpll</code> instance. The <code>scandataout</code> signal is the direct output from the scan chain shift registers. You can use this signal to read out the existing contents of the scan chain.
<code>pll_scandone</code>	No	Input signal to be driven by the <code>scandone</code> port on the <code>altpll</code> instance. This signal goes high to indicate that the reconfiguration is complete.

Table 16. *altpll_reconfig* Megafunction Output Ports

Port Name	Required	Description
<code>data_out[8..0]</code>	No	Nine-bit bus through which the parameter data can be read back by the user. The parameter value is requested using the <code>counter_type[]</code> and <code>counter_param[]</code> values, and by transitioning high the <code>read_param</code> signal, at which point the value of the parameter will be loaded from the scan chain and driven on this bus. The data will be valid once the <code>busy</code> signal is deasserted.
<code>busy</code>	No	Signal indicating that the state machine is busy either reading/writing a parameter into the scan chain, or reconfiguring the PLL. While this signal is asserted, the state machine will ignore its inputs and cannot be altered until it deasserts this signal.
<code>pll_scanclk</code>	Yes	Signal to drive the <code>scanclk</code> port on the PLL to be reconfigured.
<code>pll_scanread</code>	Yes	Signal to drive the <code>scanread</code> port on the PLL to be reconfigured.
<code>pll_scanwrite</code>	Yes	Signal to drive the <code>scanwrite</code> port on the PLL to be reconfigured.
<code>pll_scandata</code>	Yes	Signal to drive the <code>scandata</code> port on the PLL to be reconfigured.

Table 17. *altpll_reconfig* Megafunction `counter_type[3..0]` Settings

Counter Selection	<code>counter_type [3..0]</code>	
	Binary	Hexadecimal
N	0000	0x0
M	0001	0x1
CP/LF	0010	0x2
C0	0100	0x4
C1	0101	0x5
C2	0110	0x6
C3	0111	0x7
C4	1000	0x8
C5	1001	0x9
(illegal value)	All other combinations	

Table 18. altpll_reconfig Megafunction counter_param [2..0] Settings

Counter Selection	counter_param[2..0]		Width (Bits)
	Binary	Hexadecimal	
Nominal count (for enhanced M, N)	000	0x0	9
High cycles count (for enhanced C0 to C5) (1)	000	0x0	8
Low cycles count (for enhanced C0 to C5) (1)	001	0x1	8
High cycles count (for fast C0 to C3, M)	000	0x0	4
Low cycles count (for fast C0 to C3, M)	001	0x1	4
Nominal count (for fast N)	000	0x0	2
Phase step setting	010	0x2	2
Counter bypass bit	100	0x4	1
Counter odd division bit (for C0 to C5, fast M) (1)	101	0x5	1
Spread counter bypass (for enhanced M, N)	101	0x5	1
Charge pump current (for CP/LF)	000	0x0	4
Loop filter resistor (for CP/LF)	001	0x1	6
Loop filter capacitor (for CP/LF)	010	0x2	2
(illegal value)	All other combinations		

Note to Table 18:

- (1) The C4 and C5 counters are only valid for enhanced PLLs.

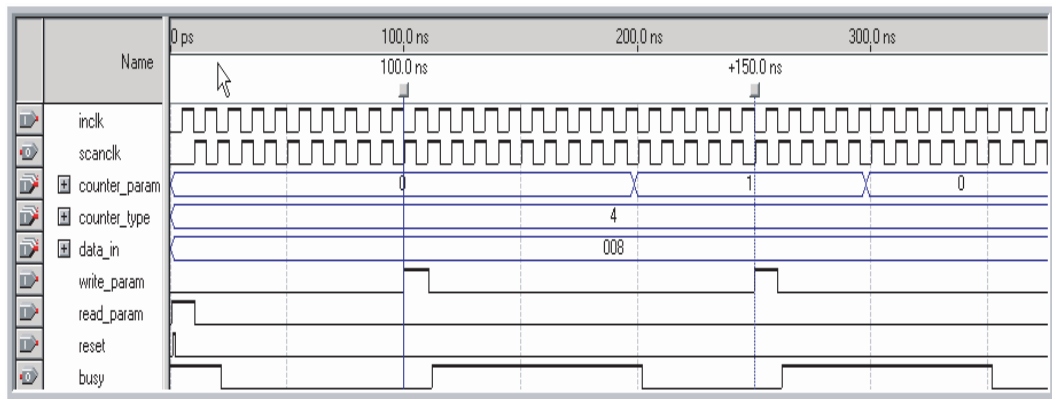
Reconfiguring the C0 Counter Using altpll_reconfig

This section describes how to reconfigure the C0 counter using the altpll_reconfig megafunction by following Steps 1 and 2.

- Reconfigure the divide-by value for counter C0 from 8 to 16. The following substeps shows how to accomplish the reconfiguration. The waveform in Figure 16 illustrates the parameter values.
 - Transition the reset signal high to initialize the state machine so that it is in a valid state before enabling the scan chain.
 - Transition the read_param signal high for one clock cycle to enable writing into the scan chain.
 - counter_type[] is set to 0x4 to select counter C0.
 - counter_param[] is set to 0x0 to select the high count parameter.

- e. `data_in[]` is set to `0x8` to specify a high count of 8.
- f. Transition the `write_param` high for one clock cycle so that at $t = 100$ ns, the data begins to load into the scan chain.
- g. The state machine transitions the `busy` signal high at the start of data loading at $t = 150$ ns until $t = 250$ ns, at which point the data loading has completed successfully.
- h. `counter_param[]` is set to `0x1` to select the low count parameter.
- i. `data_in[]` is set to `0x8` to specify a low count of 8.
- j. Transition the `write_param` high for one clock cycle so that at $t = 250$ ns, the data begins to load into the scan chain.
- k. The state machine transitions the `busy` signal high at the start of data loading at $t = 250$ ns until $t = 350$ ns, at which point the data loading has successfully completed.

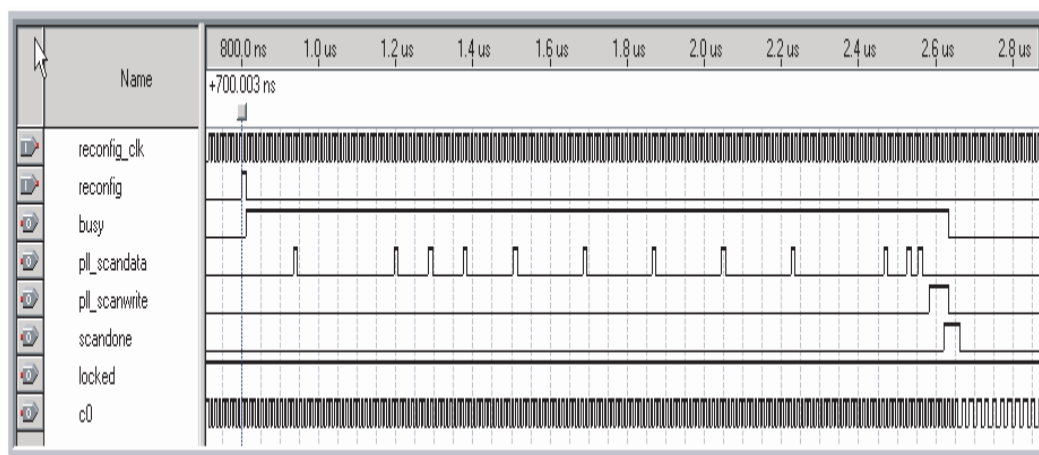
Figure 16. Waveform View of Reconfiguring the Divide-by Value for Counter C0



2. Reconfigure the PLL with current parameters in scan chain so that the output clock frequency changes from 100 MHz to 50 MHz. The following substeps show how to accomplish the reconfiguration. The waveform in [Figure 17](#) illustrates the parameter values.
 - a. Transition the `reconfig` signal high for one clock cycle so that at $t = 800$ ns, the data in the scan chain begins to load into the PLL through the PLL scan ports.

- b. The state machine transitions the `busy` signal high at the start of data loading at $t = 800 \text{ ns}$ until $t = 2.65 \mu\text{s}$, at which point the `scandata` is completely written into the scan chain.
- c. Transition the `pll_scanwrite` signal high and the new configuration data from the scan chain is loaded into the PLL. The `scandone` signal goes low at $\sim 2.75 \mu\text{s}$ to signal the successful reconfiguration of the PLL to change the C0 clock frequency from 100 MHz to 50 MHz.

Figure 17. Waveform View of Reconfiguring the PLL



You can implement phase shift stepping similarly:

1. Use `write_param` and `data_in` to write the phase control bit settings into the scan chain as shown in [Table 4 on page 11](#).
2. Toggle the `reconfig` signal the desired number of times to increment the phase of the output clock in increments of 45° or an eighth (0.125) of the VCO period. This is explained in more detail in [“Phase Shift Stepping” on page 26](#). [“Example 3: Implementing Phase Shift Stepping Using the altpll_reconfig Design With Write Parameters”](#) describes how to implement phase shift stepping.

Design Examples

This section provides examples of implementing reconfigurable PLLs in your design.

Example 1: `altpll_reconfig` Design With MIF

Unzip the `altpll_reconfig` design and compile it in the Quartus II software. The MIF has been setup for you to reconfigure the C0 counter to divide by 6 from an initial setting of divide by 12. This changes the output clock frequency from 100 to 50 MHz. To reconfigure other settings or counters, you will need to first enable the appropriate counter that you wish to reconfigure in the `altpll` MegaWizard Plug-In and then change the corresponding scan bits (as shown in [Table 12](#)) in the MIF.

Example 2: `altpll_reconfig` Design With Write Parameters

Unzip the `altpll_reconfig` design and compile it in the Quartus II software. The write parameters (`counter_type`, `counter_param`, and so on) are set up such that the output clock frequency changes from 100 MHz to 50 MHz after the PLL is reconfigured. Other changes to the PLL counters/phase shift can be made by changing the counter parameters as shown in [Table 18](#) of the “[Ports & Parameters](#)” section.

Example 3: Implementing Phase Shift Stepping Using the `altpll_reconfig` Design With Write Parameters

Unzip the `altpll_reconfig` design and compile it in the Quartus II software. The write parameters (`counter_type`, `counter_param`, and so on) are set up for stepping forward in increments of 45°, or an eighth (0.125), of the VCO period. The `reconfig` signal is toggled five times to get a phase shift of 1.25 ns. Other changes to the PLL counters/phase shift settings can be made by changing the counter parameters as shown in [Table 18](#) of the “[Ports & Parameters](#)” section.

Conclusion

PLL reconfiguration is a powerful feature that system designers can use to vary the clock output frequently and to phase shift on the fly. Important considerations such as PLL loss of lock, glitches, and output phase relationships, may affect your selections of the PLL counter and phase shift settings. Altera recommends changing PLL parameters gradually or incrementally instead of single-step drastic changes. PLL reconfiguration time is also typically less than 20 μ s, allowing you to switch rapidly between operating modes. The flexibility offered by the Stratix II PLL makes it a superior clock management system.



101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com
Applications Hotline:
(800) 800-EPLD
Literature Services:
lit_req@altera.com

Copyright © 2004 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time 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 products or services.

