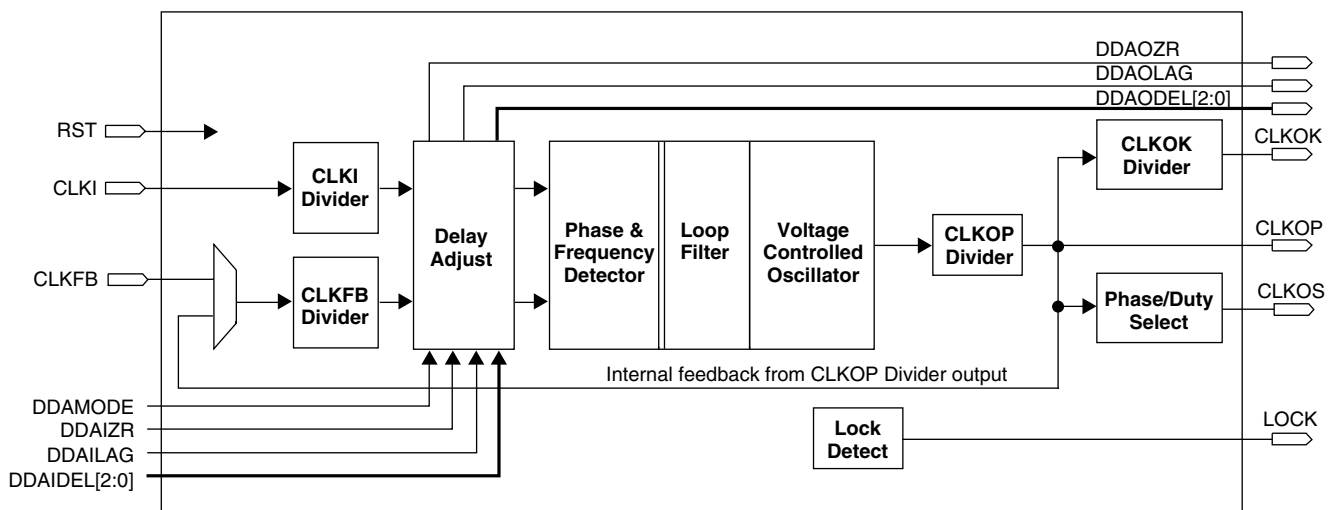


## Introduction

As clock distribution and clock skew management become critical factors in overall system performance, the Phase Locked Loop (PLL) is increasing in importance for digital designers. Lattice incorporates its sysCLOCK™ PLL technology in the LatticeECP™, LatticeEC™ and LatticeXP™ device families to help designers manage clocks within their designs. The PLL components in the LatticeECP/EC and LatticeXP device families share the same architecture. This technical note describes the features and functionalities of the PLLs and their configuration in the ispLEVER® design tool. Figure 10-1 shows the block diagram of the PLL.

**Figure 10-1. LatticeECP/EC and LatticeXP sysCLOCK PLL Block Diagram**



## Features

- Clock synthesis
- Phase shift/duty cycle selection
- Internal and external feedback
- Dynamic delay adjustment
- No external components required
- Lock detect output

## Functional Description

### PLL Divider and Delay Blocks

#### Input Clock (CLKI) Divider

The CLKI divider is used to control the input clock frequency into the PLL block. It can be set to an integer value of 1 to 16. The divider setting directly corresponds to the divisor of the output clock. The input and output of the input divider must be within the input and output frequency ranges specified in the device data sheet.

#### Feedback Loop (CLKFB) Divider

The CLKFB divider is used to divide the feedback signal. Effectively, this multiplies the output clock, because the divided feedback must speed up to match the input frequency into the PLL block. The PLL block increases the output frequency until the divided feedback frequency equals the input frequency. Like the input divider, the feedback

loop divider can be set to an integer value of 1 to 16. The input and output of the feedback divider must be within the input and output frequency ranges specified in the device data sheet.

### Delay Adjustment

The delay adjust circuit provides programmable clock delay. The programmable clock delay allows for step delays in increments of 250ps (nominal) for a total of 2.00ns lagging or leading. The time delay setting has a tolerance. See device data sheet for details. Under this mode, CLKOP, CLKOS and CLKOK are identically affected. The delay adjustment has two modes of operation:

- **Static Delay Adjustment** – In this mode, the user-selected delay is configured at power-up.
- **Dynamic Delay Adjustment (DDA)** – In this mode, a simple bus is used to configure the delay. The bus signals are available to the general purpose FPGA.

### Output Clock (CLKOP) Divider

The CLKOP divider serves the dual purposes of squaring the duty cycle of the VCO output and scaling up the VCO frequency into the 420MHz to 840MHz range to minimize jitter. Refer to Table 10-3 for CLKOP Divider value.

### CLKOK Divider

The CLKOK divider feeds the global clock net. It divides the CLKOP signal of the PLL by the value of the divider. It can be set to values of 2, 4, 6,....126,128.

## PLL Inputs and Outputs

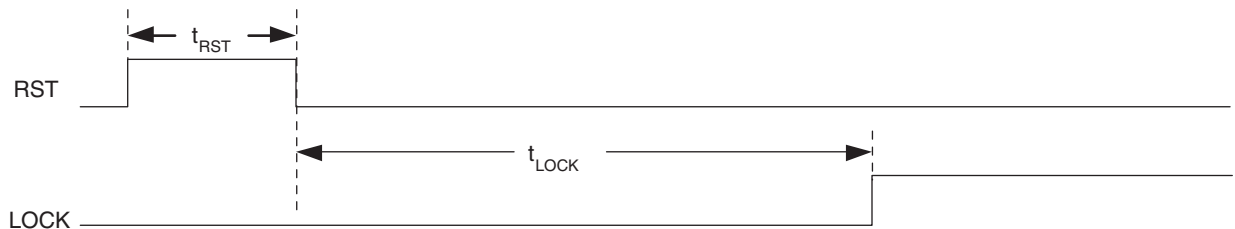
### CLKI Input

The CLKI signal is the reference clock for the PLL. It must conform to the specifications in the data sheet in order for the PLL to operate correctly. The CLKI can be derived from a dedicated dual-purpose pin or from routing.

### RST Input

The PLL reset occurs under two conditions. At power-up an internal power-up reset signal from the configuration block resets the PLL. The user controlled PLL reset signal RST is provided as part of the PLL module that can be driven by an internally generated reset function or a pin. This RST signal resets all internal PLL counters. When RST goes inactive, the PLL will start the lock-in process, and will take the  $t_{LOCK}$  time to complete the PLL lock. Figure 10-2 shows the timing diagram of the RST Input.

**Figure 10-2. RST Input Timing Diagram**



### CLKFBK Input

The feedback signal to the PLL, which is fed through the feedback divider can be derived from the global clock net, a dedicated dual-purpose pin, or directly from the CLKOP divider. Feedback must be supplied in order for the PLL to synchronize the input and output clocks. External feedback allows the designer to compensate for board-level clock alignment.

### CLKOP Output

The sysCLOCK PLL main clock output, CLKOP, is a signal available for selection as a primary clock.

### CLKOS Output with Phase and Duty Cycle Select

The sysCLOCK PLL auxiliary clock output, CLKOS, is a signal available for selection as a primary clock. The CLKOS is used when phase shift and/or duty cycle adjustment is desired. The programmable phase shift allows for

different phase in increments of 45° to 315°. The duty select feature provides duty select in 1/8th of the clock period.

### **CLKOK Output with Lower Frequency**

The CLKOK is used when a lower frequency is desired. It is a signal available for selection as a primary clock.

### **Dynamic Delay Control I/O Ports (for EHXPLL Only)**

Refer to Table 10-2 and page 10-6 for detailed information.

### **LOCK Output**

The LOCK output provides information about the status of the PLL. After the device is powered up and the input clock is valid, the PLL will achieve lock within the specified lock time. Once lock is achieved, the PLL lock signal will be asserted. If, during operation, the input clock or feedback signals to the PLL become invalid, the PLL will lose lock. PLL RST must be applied to re-synchronize the PLL to the reference clock. The LOCK signal is available to the FPGA routing to implement generation of RST.

### **PLL Attributes**

The PLL utilizes several attributes that allow the configuration of the PLL through source constraints. The following section details these attributes and their usage.

#### **FIN**

The input frequency can be any value within the specified frequency range based on the divider settings.

#### **CLKI\_DIV, CLKFB\_DIV, CLKOP\_DIV, CLKOK\_DIV**

These dividers determine the output frequencies of each output clock. The user is not allowed to input an invalid combination; determined by the input frequency, the dividers, and the PLL specifications.

#### **CLKOP\_FREQ, CLKOK\_FREQ**

These output clock frequencies determine the divider values.

#### **FDEL**

The FDEL attribute is used to pass the Delay Adjustment step associated with the Output Clock of the PLL. This allows the user to advance or retard the Output Clock by the step value passed multiplied by 250ps(nominal). The step ranges from -8 to +8 resulting the total delay range to +/- 2ns.

#### **PHASEADJ**

The PHASEADJ attribute is used to select Coarse Phase Shift for CLKOS output. The phase adjustment is programmable in 45° increments.

#### **DUTY**

The DUTY attribute is used to select the Duty Cycle for CLKOS output. The Duty Cycle is programmable at 1/8 of the period increment.

#### **FB\_MODE**

There are three sources of feedback signals that can drive the CLKFB Divider: internal, clocktree and external feedback. Clocktree feedback is used by default. Internal feedback takes the CLKOP output at CLKOP Divider output before the Clocktree to minimize the feedback path delay. The external feedback is driven from the pin.

#### **DELAY\_CNTL**

This attribute is designed to select the Delay Adjustment mode. If the attribute is set to "DYNAMIC" the delay control switches between Dynamic and Static depending upon the input logic of DDAMODE pin. If the attribute is set to "STATIC", Dynamic Delay inputs are ignored in this mode.

## LatticeECP/EC and LatticeXP PLL Primitive Definitions

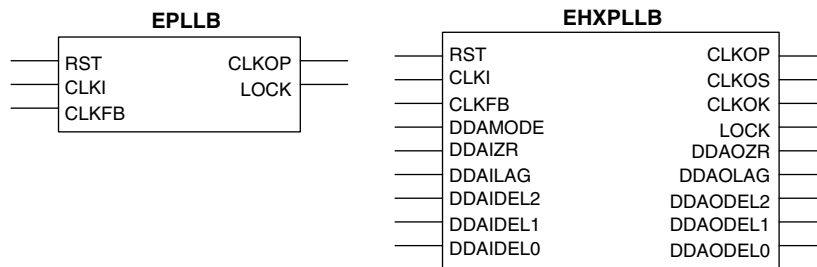
Two PLL primitives may be used for LatticeECP/EC and LatticeXP PLL implementation. Users can choose either primitive depending on the design requirement of the PLL. The definitions of the PLL I/O ports are shown in Table 10-2. Some of the features are optional as shown in the table below.

Figure 10-3 shows the LatticeECP/EC and LatticeXP PLL primitives library symbols and Table 10-1 lists the signals.

The EPLLB is the most commonly used module of the PLL. In this primitive, the CLKOS is not provided. Instead, the CLKOP has more divider setting values to maximize the  $F_{VCO}$  for full PLL performance.

The EHXPLLB includes all features available in the PLL including Dynamic Delay Adjustment. Some of the features are optional, as shown in Table 10-1.

**Figure 10-3. LatticeECP/EC and LatticeXP PLL Primitive Symbols**



**Table 10-1. Signal Usage in EPLLB and EHXPLLB Primitives**

Signal	EPLLB	EHXPLLB	Optional
CLKI	X	X	No
CLKFB	X	X	No
CLKOP	X	X	Yes <sup>1</sup>
CLKOS	—	X	Yes <sup>1</sup>
CLKOK	—	X	Yes <sup>1</sup>
LOCK	X	X	Yes
RST	X	X	Yes
DDAMODE	—	X	Yes
DDAIZR	—	X	Yes
DDAILAG	—	X	Yes
DDAIDEL(0:2)	—	X	Yes
DDAOZR	—	X	Yes
DDAOLAG	—	X	Yes
DDAODEL(0:2)	—	X	Yes

1. At least one of these output clocks must be used.

**Table 10-2. LatticeECP/EC and LatticeXP PLL I/O Definitions**

Signal	I/O	Description	Freq. (MHz) <sup>2</sup>
CLKI	I	PLL clock input from either internal logic or dedicated clock input pin.	—
CLKFB <sup>1</sup>	I	Feedback clock input from either internal node, internal feedback from CLKOP or dedicated external feedback pin.	—
RST	I	“1” to Reset PLL.	—
CLKOP	O	PLL output clock to clock tree (no phase shift).	—
CLKOS	O	PLL output clock to clock tree (programmable phase shift/duty cycle).	—
CLKOK	O	PLL output to clock tree (through K-divider for lower frequency clock).	—
LOCK	O	“1” indicates PLL locked to CLKI.	—
DDAMODE	I	DDA Mode. “1” Pin control (dynamic), “0”: Fuse Control (static).	—
DDAIZR	I	DDA Delay Zero. “1”: delay = 0, “0”: delay = [DDILAG + DDAIDEL].	—
DDAILAG	I	DDA Lag/Lead. “1”: Lead, “0”: Lag.	—
DDAIDEL[2:0]	I	DDA Delay	—
DDAOZR	O	DDA Delay Zero Output	—
DDAOLAG	O	DDA Lag/Lead Output	—
DDAODEL[2:0]	O	DDA Delay Output	—

1. For internal feedback, user does not specify CLKFB and software implements internal feedback automatically.
2. Please refer to data sheet for latest data.

## PLL Attributes Definitions

Both EPLL and EHXPLL can be configured through attributes in the source code. The following section details these attributes and their usage.

**Table 10-3. LatticeECP/EC and LatticeXP PLL Attributes**

Parameter	Description	Value	EPLL	EHXPLL	Default
FIN	Input (CLKI) Frequency (MHz)	See note 5.	YES	YES	100
CLKOP_FREQ	CLKOP Frequency (MHz)	See note 5.	YES	YES	100
CLKOK_FREQ	CLKOK Frequency (MHz)	See note 5.	—	YES	50
CLKI_DIV <sup>1</sup>	CLKI Divider Setting	1 to 16	YES	YES	1
CLKFB_DIV	CLKFB Divider Setting	1 to 16	YES	YES	1
CLKOP_DIV	CLKOP Divider Setting	(Note 2)	2,4,6,8,10,... 30,32	2,4,8, 16, 32	8 <sup>3</sup>
CLKOK_DIV	CLKOK Divider Setting	2,4,6,...,126,128	—	YES	2
FDEL	Fine Delay Adjust	-8 to 8	YES	YES	0
PHASEADJ	Coarse Phase Shift Selection (°)	0, 45, 90...315	—	YES	0
DUTY	Duty Cycle Selection (1/8 increment)	1 to 7	—	YES	4
DELAY_CNTL <sup>4</sup>	Delay Control	DYNAMIC/STATIC	STATIC	YES	STATIC
FB_MODE	Feedback Mode	INTERNAL/CLOCK TREE/EXTERNAL	YES	YES	CLOCK TREE

1. All divider settings are user-transparent in Normal Mode. These are user attributes in Advanced Mode.
2. The CLKOP Divider values are 2,4,6,8,...30,32 for primitive EPLL and 2,4,8,16,32 for EHXPLL.
3. CLKOP\_DIV value must be calculated to maximize the  $f_{VCO}$  within the specified range based on FIN, CLKOP\_FREQ and CLKOK\_FREQ in conjunction with CLKI\_DIV and CLKFB\_DIV values.
4. DYNAMIC: This mode can switch delay control between Dynamic and Static depending upon the input logic of DDAMODE pin.  
STATIC: This is Static Control only mode. All DDA input ports are ignored in this mode.
5. Please refer to data sheet for latest data.

**Dynamic Delay Adjustment (for EHXPLL only)**

The Dynamic Delay Adjustment is controlled by the DDAMODE input. This feature is available in the EHXPLL primitive only. When the DDAMODE input is set to “1”, the delay control is done through the inputs, DDAIZR, DDAILAG and DDAIDEL(2:0). For this mode, the attribute “DELAY\_CNTL” must be set to “DYNAMIC”. Table 10-4 shows the delay adjustment values based on the attribute/input settings.

In this mode, the PLL may come out of lock due to the abrupt change of phase. RST must be asserted to re-lock the PLL. Upon de-assertion of RST, the PLL will start the lock-in process and will take the  $t_{LOCK}$  time to complete the PLL lock.

**Table 10-4. Delay Adjustment**

DDAMODE = 1: Dynamic Delay Adjustment			DELAY 1 $t_{DLY}$ = 250ps (nominal)	DDAMODE = 0
DDAIZR	DDAILAG	DDAIDEL[2:0]		Equivalent FDEL Value
0	1	111	Lead 8 $t_{DLY}$	-8
0	1	110	Lead 7 $t_{DLY}$	-7
0	1	101	Lead 6 $t_{DLY}$	-6
0	1	100	Lead 5 $t_{DLY}$	-5
0	1	011	Lead 4 $t_{DLY}$	-4
0	1	010	Lead 3 $t_{DLY}$	-3
0	1	001	Lead 2 $t_{DLY}$	-2
0	1	000	Lead 1 $t_{DLY}$	-1
1	Don't Care	Don't Care	No delay	0
0	0	000	Lag 1 $t_{DLY}$	1
0	0	001	Lag 2 $t_{DLY}$	2
0	0	010	Lag 3 $t_{DLY}$	3
0	0	011	Lag 4 $t_{DLY}$	4
0	0	100	Lag 5 $t_{DLY}$	5
0	0	101	Lag 6 $t_{DLY}$	6
0	0	110	Lag 7 $t_{DLY}$	7
0	0	111	Lag 8 $t_{DLY}$	8

Note:  $t_{DLY}$  = Unit Delay Time = 250 ps (nominal). See the data sheet for the tolerance of this delay

## PLL Usage in Module Manager and HDL

### Including sysCLOCK PLLs in a Design

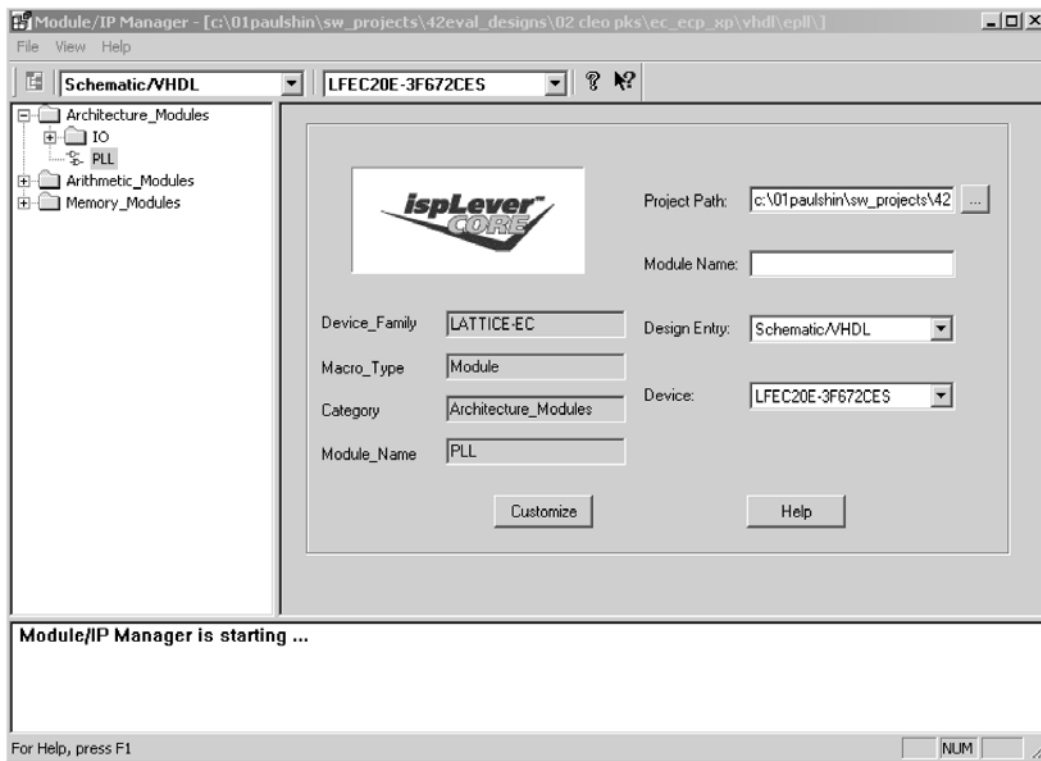
The sysCLOCK PLL capability can be accessed either through the Module Manager or directly instantiated in a design's source code. The following sections describe both methods.

### Module Manager Usage

The LatticeECP/EC and LatticeXP PLL is fully supported in Module Manager in the ispLEVER software. The Module Manager allows the user to define the desired PLL using a simple, easy-to-use GUI. Following definition, a VHDL or Verilog module that instantiates the desired PLL is created. This module can be included directly in the user's design.

Figure 10-4 shows the main window when PLL is selected. The only entry required in this window is the module name. After entering the module name, clicking on "Customize" will open the "General Options" window as shown in Figure 10-5.

**Figure 10-4. Module IP Manager Main Window**



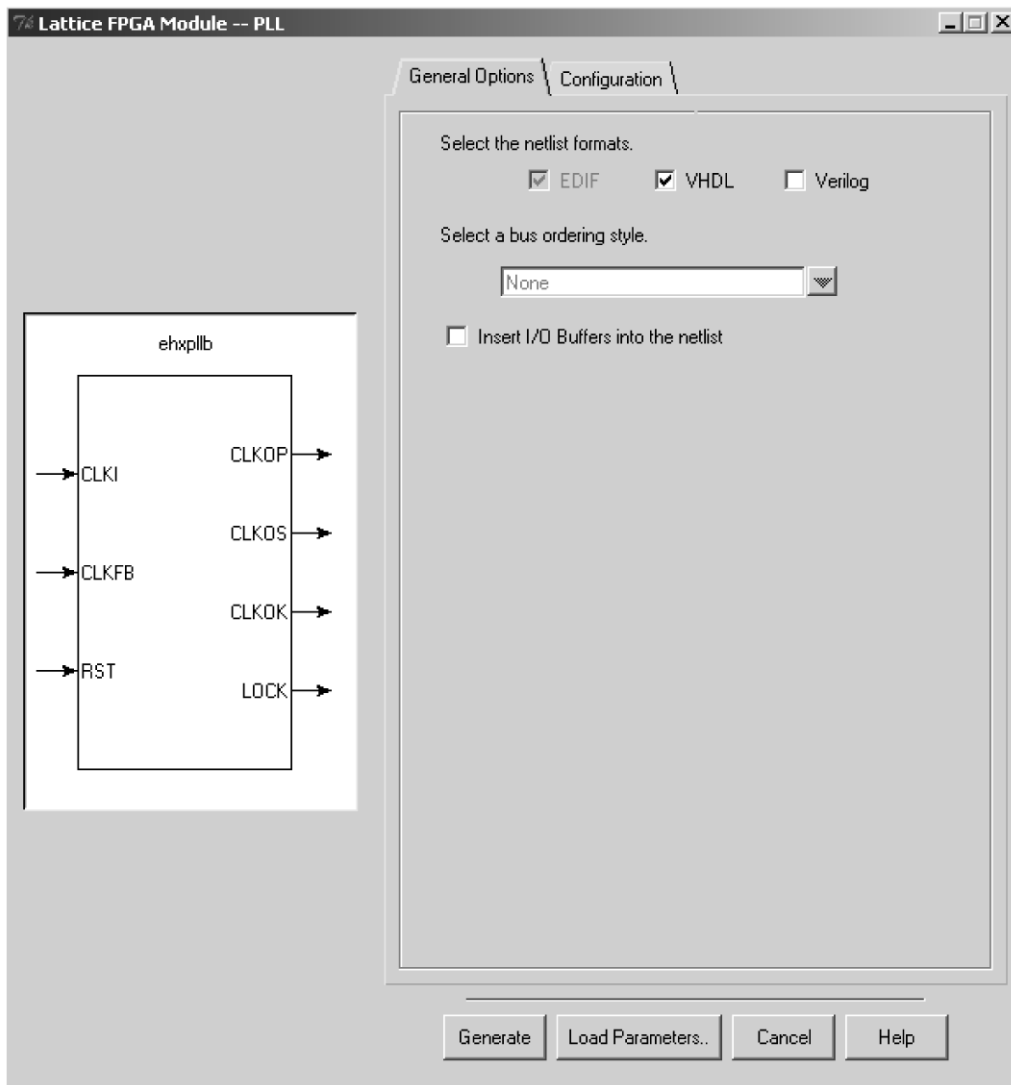
**General Options Tab**

The General Options Tab provides the ability to define the following:

- Netlist format type: EDIF, VHDL or Verilog
- Synthesizer: Synplicity or Exemplar

Clicking 'Generate' creates a VHDL (module name.vhd) or Verilog (module name.v) file in the working directory that instantiates the core. At the same time a parameter file (module name.lpc) file is created in the working directory. The load parameters button can be used to reload configurations from previously created parameter files (\*.lpc files).

**Figure 10-5. General Options Tab**



The General Options Tab also provides the ability to define the following:

- Bus ordering style
- Insertion of I/O Buffers in the netlist

These options are reserved for future use.

**Configuration Tab**

The Configuration Tab lists all user accessible attributes. Default values are set initially. Table 10-5 describes the user accessible attributes and their usage.

**Table 10-5. User Accessible Attributes**

GUI Text	Description	Values, Ranges	Normal	Advanced
PLL Type	PLL Primitive type	EPLL, EHXPLL	Yes	Yes
Mode	PLL Configuration mode	Normal, Advanced	—	—
CLKI Frequency	Input Clock Frequency	See note below.	Yes	Yes
CLKOP Frequency	Output Clock (CLKOP) Frequency	See note below.	Yes	No
CLKOK Frequency	Output Clock (CLKOK) Frequency	See note below.	Yes	No
N(CLKFB_DIV)	Feedback Clock Divider	1 to 16	No	Yes
M(CLKI_DIV)	CLKI Divider	1 to 16	No	Yes
V(CLKOP_DIV)	CLKOP Divider	EPLL: 2, 4, 6,...30, 32 EHXPLL: 2, 4, 8, 16, 32	No	Yes
K(CLKOK_DIV)	CLKOK Divider	2, 4, 6,...126, 128	No	Yes
Fine Delay Adjust	Delay Adjustment steps: 250pS/step (nominal)	-8, -7,...0,..7, 8	Yes	Yes
Feedback Mode	Feedback Mode	Internal/Clocktree/External	Yes	Yes
Delay Control	Delay Control mode	Dynamic/Static	Yes	Yes
PhaseAdj	CLKOS Phase Shift Selection	0, 45, 90, 135, 180, 225, 270, 315	Yes	Yes
Duty Cycle Selection	Duty Cycle Selection (1/8 increment)	1 to 7	Yes	Yes

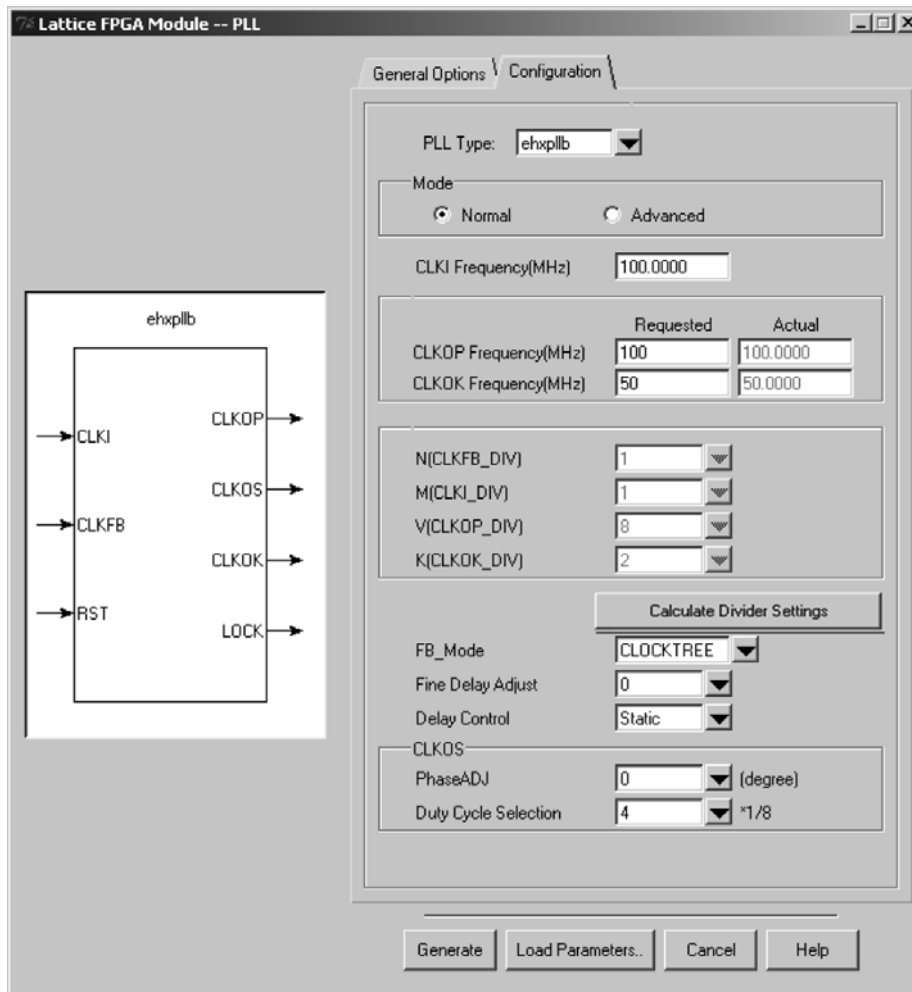
Note: Please refer to data sheet for latest data.

**Mode**

There are two modes in the Configuration Tab which can be used to configure the PLL, Normal and Advanced.

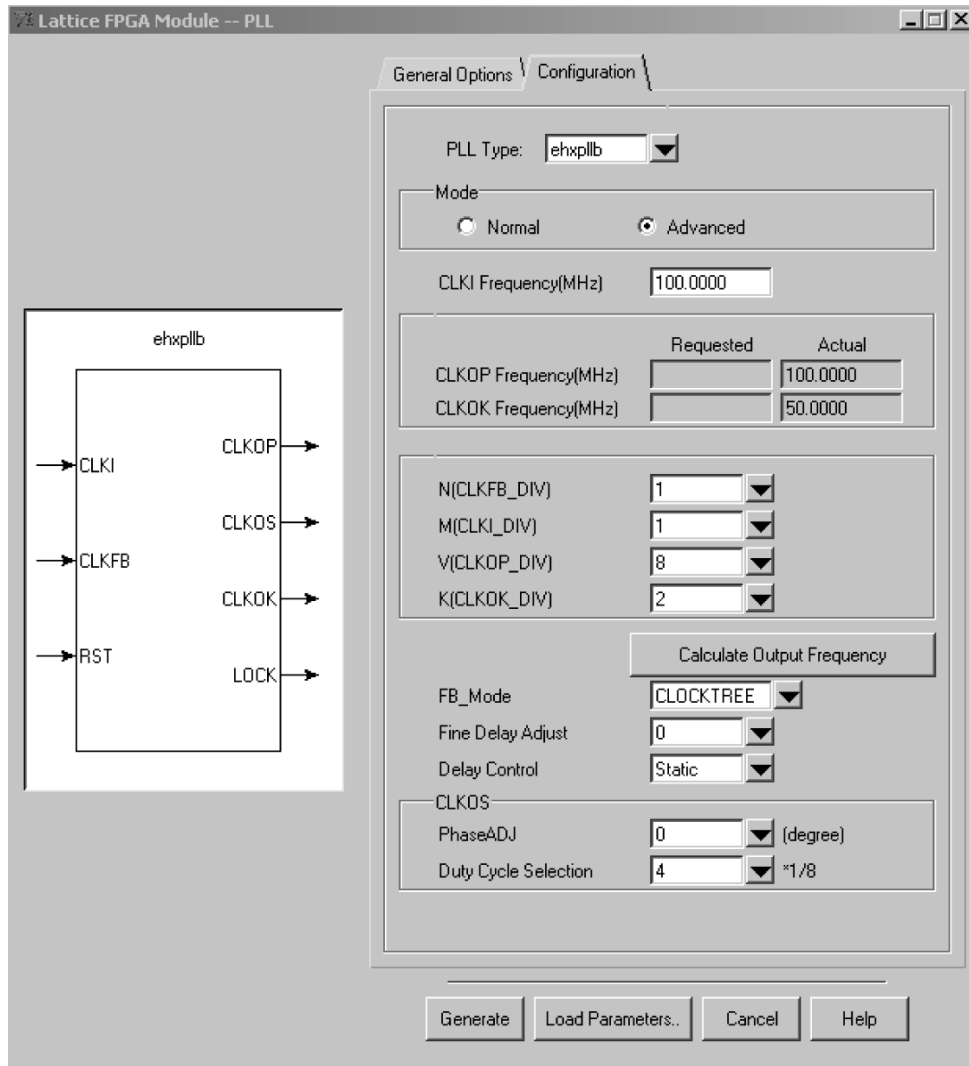
**Normal Mode:** In this mode, the user enters input and output clock frequencies and the software calculates the divider settings for the user. If the output frequency the user entered is not achievable, the nearest frequency will be displayed in the 'Actual' text box. After input and output frequencies are entered, clicking the 'Calculate Divider Settings' button will display the divider values. Figure 10-6 shows the Normal mode configuration tab.

*Figure 10-6. Configuration Tab in Normal Mode*



**Advanced Mode:** In this mode, user sets input frequency and divider settings. It is assumed the user is familiar with the PLL operation. The user must choose the V values to maximize the  $f_{VCO}$  to achieve optimum PLL performance. After input frequency and divider settings are set, clicking the ‘Calculate Output Frequency’ button will display the frequencies. Figure 10-7 shows the Advanced mode configuration tab. The figure also illustrates that when ‘Dynamic’ Delay Control is selected, the EHXPLL symbol diagram adds Dynamic Delay control signals.

Figure 10-7. Configuration Tab in Advanced Mode



## Equations for Generating Input and Output Frequency Ranges

The values of  $f_{IN}$ ,  $f_{OUT}$  and  $f_{VCO}$  are the absolute frequency ranges for the PLL. The values of  $f_{INMIN}$ ,  $f_{INMAX}$ ,  $f_{OUTMIN}$  and  $f_{OUTMAX}$  are the calculated frequency ranges based on the divider settings. These calculated frequency ranges become the limits for the specific divider settings used in the design.

**Table 10-6. Frequency Limits**

Parameter	Description	Values, Ranges
$f_{IN}$	CLKI, CLKFB Frequency	See note below.
$f_{OUT}$	CLKOP, CLKOS Frequency	See note below.
$f_{OUTK}$	CLKOK Frequency	See note below.
$f_{VCO}$	VCO Operating Frequency	See note below.
CLKI Divider	Input Clock Divider	1 to 16
CLKFB Divider	Feedback Clock Divider	1 to 16
CLKOP Divider	Output Clock Divider	EPLL: 2, 4, 6...30, 32 EHXPLL: 2, 4, 8, 16, 32
CLKOK Divider	CLKOK Output Clock Divider	2,4,6,8,...,126,128
Maximum (N * V)	CLKFB_DIV * CLKOP_DIV	32
$f_{PFD}$ ( $f_{IN} / M$ ) (Hz)	PFD input Frequency	See note below.

Note: Please refer to data sheet for latest data.

The divider names are abbreviated with legacy names as:

- CLKI DIVIDER:M
- CLKFB DIVIDER:N
- CLKOP DIVIDER:V
- CLKOK DIVIDER:K

for use in the equations below.

Please refer to Figure 10-1 and Table 10-6 for the discussion below.

### $f_{VCO}$ Constraint

From the loop:

$$f_{OUT} = f_{IN} * (N/M) \quad (1)$$

From the loop:

$$f_{VCO} = f_{OUT} * V \quad (2)$$

Substitute (1) in (2) yields:

$$f_{VCO} = f_{IN} * (N/M) * V \quad (3)$$

Arrange (3):

$$f_{IN} = (f_{VCO} / (V*N)) * M \quad (4)$$

From equation (4):

$$f_{INMIN} = ((f_{VCOMIN} / (V*N)) * M) \quad (5)$$

$$f_{INMAX} = (f_{VCOMAX} / (V*N)) * M \quad (6)$$

### $f_{PFD}$ Constraint

From the loop:

$$f_{PFD} = f_{IN} / M \quad (7)$$

$$f_{IN} = f_{PFD} * M$$

$$f_{INMIN} = f_{PFDMIN} * M = 25 * M \text{ (assume } f_{PFDMIN} = 25) \quad (8)$$

Equation (5) becomes:

$$f_{INMIN} = ((f_{VCOMIN} / (V*N))*M, \text{ if below } 25 * M \text{ round up to } 25 * M) \quad (9)$$

From the loop:

$$f_{INMAX} = f_{PFDMAX} * M = 420 * M \quad (10)$$

Assume  $f_{INMAX} = 420$

Equation (6) becomes:

$$f_{INMAX} = (f_{VCOMAX} / (V*N))*M, \text{ if above } 420 \text{ round down to } 420 \quad (11)$$

From equation (1):

$$f_{OUTMIN} = f_{INMIN} * (N/M), \text{ if below } 25 * N \text{ round up to } 25 * N \quad (12)$$

$$f_{OUTMAX} = f_{INMAX} * (N/M), \text{ if above } 420 \text{ round down to } 420 \quad (13)$$

$$f_{OUTKMIN} = f_{OUTMIN} / K$$

$$f_{OUTKMAX} = f_{OUTMAX} / K$$

### Example (EPLL B)

Assume:

$$f_{IN} = 40\text{MHz}, M = 2, N = 3, V = 4$$

Then:

$$f_{OUT} = 40 * 3 / 2 = 60 \quad (\text{within range})$$

$$f_{VCO} = 60 * 4 = 240 \quad (\text{out of range})$$

$$f_{PFD} = 40 / 2 = 20 \text{ or } 60 / 3 = 20 \quad (\text{out of range})$$

Let's assume  $M=1$ . Then:

$$f_{OUT} = 40 * 3 / 1 = 120 \quad (\text{within range})$$

$$f_{VCO} = 120 * 4 = 480 \quad (\text{within range but not optimum frequency})$$

$$f_{PFD} = 40 / 1 \text{ or } 120 / 3 = 40 \quad (\text{within range})$$

In this case,  $V=6$  will satisfy all conditions.

## Direct Instantiation Into Source Code

If desired, the Module Manager can be bypassed and the sysCLOCK PLL instantiated directly in the source code. Appendix A provides examples of source code generated by the Module Manager. These examples can be used as templates for directly instantiating the sysCLOCK PLL in the source code.

## Other Design Considerations

### Jitter Considerations

The Clock Output jitter specifications assume that the reference clock is free of jitter. Even if the clock source is clean, there are a number of sources that would place noise in the PLL clock input. While intrinsic jitter is not avoidable, there are ways to minimize the input jitter and output jitter.

Signal inputs that share the same I/O bank with PLL clock inputs are preferably less noisy inputs and slower switching signals. Minimize placing any high speed and noisy signals in the same I/O bank with clock signals if possible. Use differential signaling if applicable.

When external feedback is used, the PCB path must be well designed to avoid reflection as well as noise coupling from adjacent signal sources. A shorter PCB feedback path length does not necessarily reduce the feedback input jitter.

### Simulation Limitations

- The simulation does not compensate for external delays and dividers in the feedback loop.
- The LOCK signal is not simulated according to the  $t_{LOCK}$  specification. The LOCK signal will appear active shortly after the simulation begins, but will remain active throughout the simulation.
- The jitter specifications are not included.

### Dynamic Clock Selection (DCS)

DCS is a global clock buffer incorporating a smart multiplexer function that takes two independent input clock sources and avoids glitches or runt pulses on the output clock, regardless of when the enable signal is toggled. The DCS blocks are located in pairs at the center of each side of the device. Thus, there are eight of them in every device.

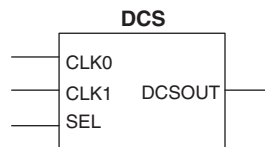
**Table 10-7. DCS I/O**

I/O	Name	Description
Input	SEL	Input Clock Select
	CLK0	Primary Clock Input 0
	CLK1	Primary Clock Input 1
Output	DCSOUT	To Primary Clock

**Table 10-8. DCS Attributes**

Attribute Name	Description	Output		Value
		SEL=0	SEL=1	
DCS MODE	Rising edge triggered, latched state is high	CLK0	CLK1	POS (Default)
	Falling edge triggered, latched state is low	CLK0	CLK1	NEG
	Sel is active high, Disabled output is low	0	CLK1	HIGH_LOW
	Sel is active high, Disabled output is high	1	CLK1	HIGH_HIGH
	Sel is active low, Disabled output is low	CLK0	0	LOW_LOW
	Sel is active low, Disabled output is high	CLK0	1	LOW_HIGH
	Buffer for CLK0	CLK0	CLK0	CLK0
	Buffer for CLK1	CLK1	CLK1	CLK1

**Figure 10-8. DCS Primitive Symbol**



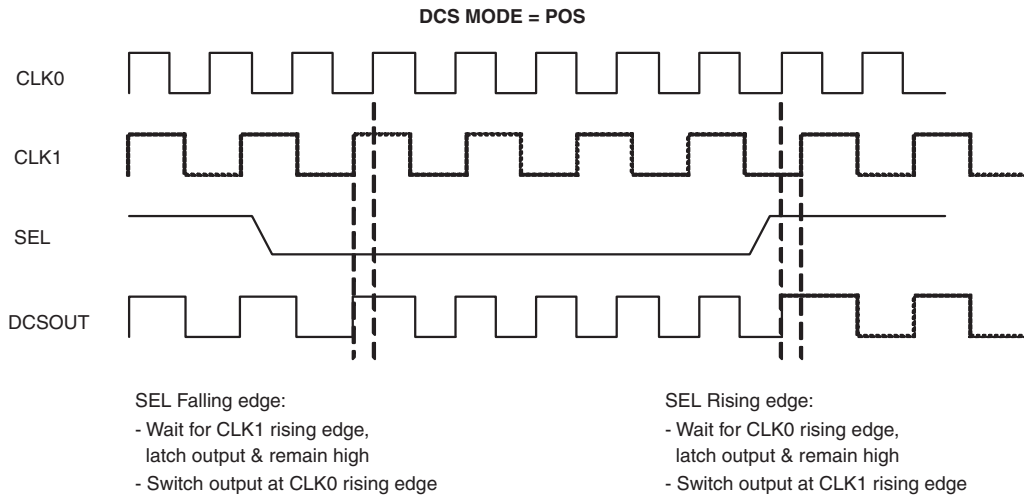
**DCS Waveforms**

The DCSOUT waveform timing is described in Figure 10-9 for each mode. The ‘POS’ and ‘NEG’ modes describe DCSOUT timing at both the falling and rising edges of SEL.

*Figure 10-9. DCS Waveforms*

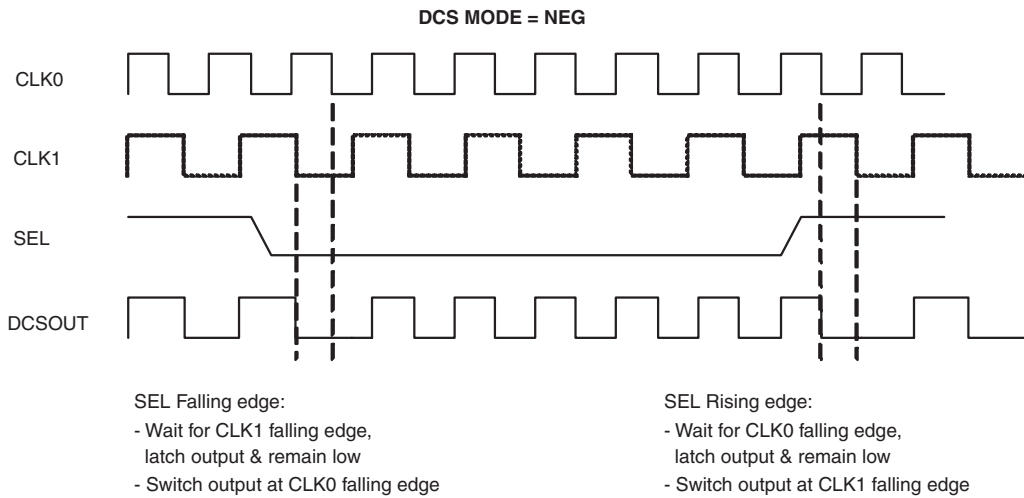
**DCS MODE = POS**

At the rising edge (POS) of SEL, the DCSOUT changes from CLK0 to CLK1. This mode is the default mode.



**DCS MODE = NEG**

At the falling edge (NEG) of SEL, the DCSOUT changes from CLK0 to CLK1.

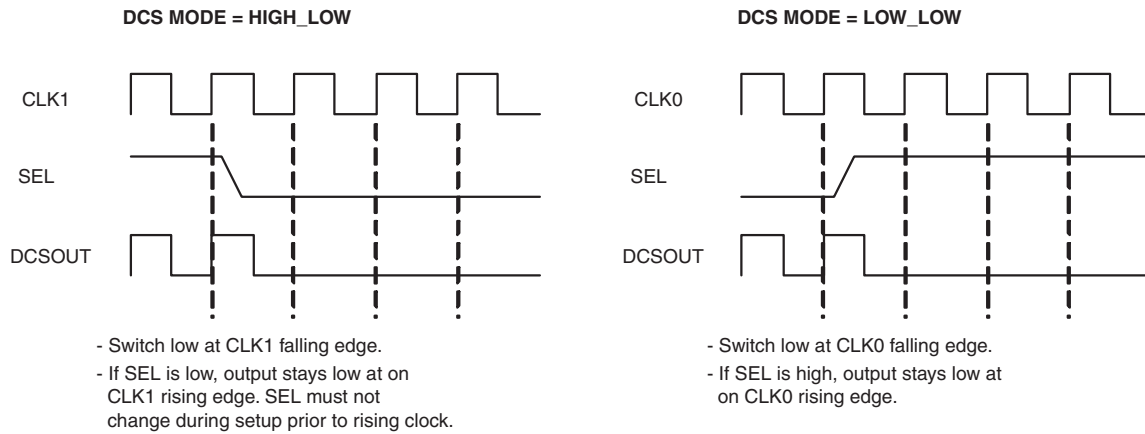


**DCS MODE = HIGH\_LOW**

SEL is active high (HIGH) to select CLK1, and the disabled output is LOW.

**DCS MODE = LOW\_LOW**

SEL is active low (LOW) to select CLK0, and the disabled output is LOW.

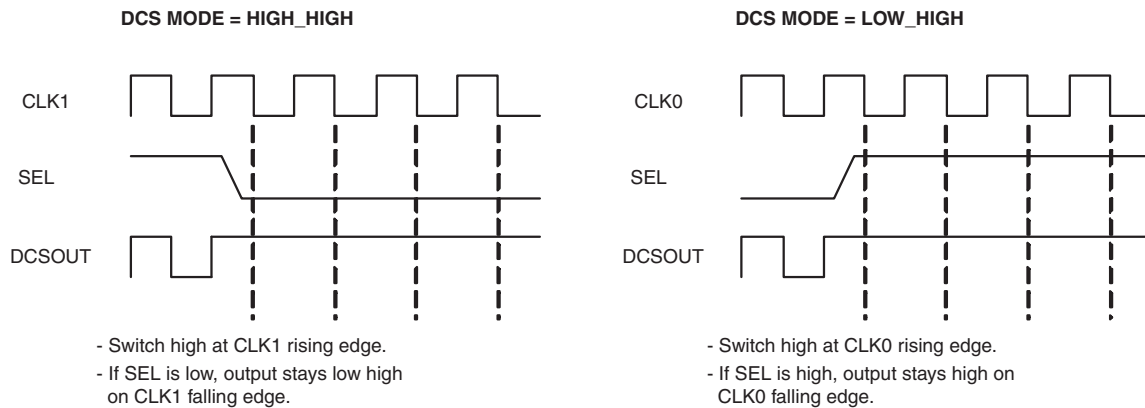


**DCS MODE = HIGH\_HIGH**

SEL is active high (HIGH) to select CLK1, and the disabled output is HIGH.

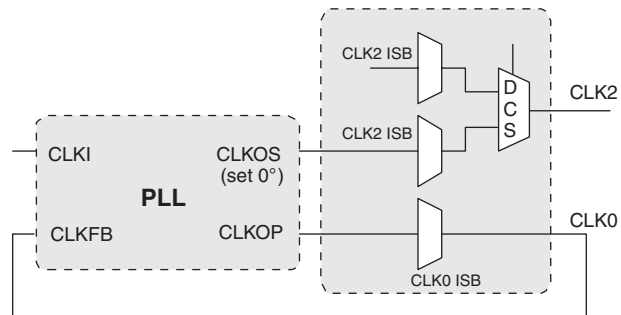
**DCS MODE = LOW\_HIGH**

SEL is active low (LOW) to select CLK0, and the disabled output is HIGH.



**Use of DCS with PLL**

The four PLL CLKOP sources reach CLK0 and CLK1 of the quadrant clock. When using the DCS, the PLL needs a free-running feedback path to keep the PLL in lock. The user should use CLKOP as this feedback path, and CLKOS as the input into the DCS. CLKOP does not reach CLK2 or CLK3 to prevent the user from using the PLL improperly with DCS. See Figure 10-10.

**Figure 10-10. Implementation of Dynamic Clock Select for a PLL Clock (Must Use Both CLKOP and CLKOS)**

## Technical Support Assistance

Hotline: 1-800-LATTICE (North America)  
+1-408-826-6002 (Outside North America)  
e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)  
Internet: [www.latticesemi.com](http://www.latticesemi.com)

---

## Appendix A. Source Code Examples

### DCS Usage with Verilog

```

module dcs(clk0,clk1,sel,dcsout);

input clk0, clk1, sel;
output dcsout;

DCS DCSInst0 (.SEL(sel),.CLK0(clk0),.CLK1(clk1),.DCSOUT(dcsout));
defparam DCSInst0.DCSMODE = "CLK0";

endmodule

```

### DCS Usage with VHDL

```

COMPONENT DCS
-- synthesis translate_off
    GENERIC
        (
            DCSMODE : string := "POS"
        );
-- synthesis translate_on

    PORT
        (
            CLK0      :IN   std_logic;
            CLK1      :IN   std_logic;
            SEL        :IN   std_logic;
            DCSOUT    :OUT  std_logic
        );
END COMPONENT;

attribute DCSMODE : string;
attribute DCSMODE of DCSInst0 : label is "POS";

begin

DCSInst0: DCS
-- synthesis translate_off

    GENERIC MAP(
        DCSMODE => "POS"
    )
-- synthesis translate_on

    PORT MAP
        (
            SEL        => clksel,
            CLK0       => dcsclock0,
            CLK1       => sysclk1,
            DCSOUT     => dcsclock
        );

```

---

## PLL Source Code Examples Generated by Module Manager

### EPLL Module (Verilog)

```

`timescale 1 ns / 100 ps
module ep11 (RST, CLKI, CLKFB, CLKOP, LOCK);

input  RST;
input  CLKI;
input  CLKFB;
output CLKOP;
output LOCK;

defparam P11Inst0.FIN = "33";
defparam P11Inst0.CLKI_DIV = "1";
defparam P11Inst0.CLKOP_DIV = "16";
defparam P11Inst0.CLKFB_DIV = "2";
defparam P11Inst0.FDEL = "-5";
defparam P11Inst0.WAKE_ON_LOCK = "OFF";
defparam P11Inst0.FB_MODE = "CLOCKTREE";
EPLL P11Inst0(.LOCK(LOCK), .CLKOP(CLKOP), .CLKFB(CLKFB), .CLKI(CLKI));

endmodule

```

### EPLL Module (VHDL)

```

--VHDL netlist generated by scuba
--timeStamp 2004 5 06 10 59 34
USE STD.TEXTIO.ALL;
Library IEEE, EC;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_arith.ALL;
USE EC.COMPONENTS.ALL;
USE WORK.ALL;
ENTITY ep11 IS

    PORT (
        RST           :IN    std_logic;
        CLKI          :IN    std_logic;
        CLKFB         :IN    std_logic;
        CLKOP         :OUT   std_logic;
        LOCK          :OUT   std_logic);

END ep11;

ARCHITECTURE V OF ep11 IS
    COMPONENT EPLL
        GENERIC (
            FIN : string := "33";
            CLKI_DIV : string := "1";
            CLKOP_DIV : string := "12";
            CLKFB_DIV : string := "2";
            FDEL : string := "0";
            FB_MODE : string := "CLOCKTREE";
            WAKE_ON_LOCK : string := "ON");
        PORT (
            RST           :IN    std_logic;
            CLKI          :IN    std_logic;
            CLKFB         :IN    std_logic;
            CLKOP         :OUT   std_logic;
            LOCK          :OUT   std_logic);
    END COMPONENT;

```

```

BEGIN
  PllInst0: EPLL
    GENERIC MAP (
      FIN => "33",
      CLKI_DIV => "1",
      CLKOP_DIV => "12",
      CLKFB_DIV => "2",
      FDEL => "0",
      FB_MODE => "CLOCKTREE";
      WAKE_ON_LOCK => "ON")
    PORT MAP (
      RST          => RST,
      LOCK        => LOCK,
      CLKOP       => CLKOP,
      CLKFB       => CLKFB,
      CLKI        => CLKI);
END V;

```

## EHXPLL Module (Verilog)

```

`timescale 1 ns / 100 ps
module pllX( CLKI, CLKFB, RST, DDAMODE, DDAIZR, DDAILAG, DDAIDEL0, DDAIDEL1,
            DDAIDEL2, CLKOP, CLKOS, CLKOK, LOCK, DDAOZR, DDAOLAG, DDAODEL0,
            DDAODEL1, DDAODEL2);

input  CLKI, CLKFB, RST, DDAMODE, DDAIZR, DDAILAG, DDAIDEL0, DDAIDEL1, DDAIDEL2;
output CLKOP, CLKOS, CLKOK, LOCK, DDAOZR, DDAOLAG, DDAODEL0, DDAODEL1, DDAODEL2;

defparam PllInst0.FIN = "100";
defparam PllInst0.CLKI_DIV = "1";
defparam PllInst0.CLKOP_DIV = "2";
defparam PllInst0.CLKFB_DIV = "1";
defparam PllInst0.FDEL = "0";
defparam PllInst0.WAKE_ON_LOCK = "OFF";
defparam PllInst0.FB_MODE = "CLOCKTREE";
defparam PllInst0.CLKOK_DIV = "2";
defparam PllInst0.PHASEADJ = "270";
defparam PllInst0.DUTY = "4";
defparam PllInst0.DELAY_CNTL = "STATIC";

EHXPLL PllInst0(.DDAODEL2(DDAODEL2), .DDAODEL1(DDAODEL1), .DDAODEL0(DDAODEL0),
               .DDAOLAG(DDAOLAG), .DDAOZR(DDAOZR), .LOCK(LOCK), .CLKOK(CLKOK),
               .CLKOS(CLKOS), .CLKOP(CLKOP), .DDAIDEL2(DDAIDEL2), .DDAIDEL1(DDAIDEL1),
               .DDAIDEL0(DDAIDEL0), .DDAILAG(DDAILAG), .DDAIZR(DDAIZR),
               .DDAMODE(DDAMODE), .RST(RST), .CLKFB(CLKFB), .CLKI(CLKI));

endmodule

```

## EHXPLL Module (VHDL)

```

--VHDL netlist generated by scuba
--timeStamp 2004 3 24 11 14 21
USE STD.TEXTIO.ALL;
Library IEEE, EC;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_arith.ALL;
USE EC.COMPONENTS.ALL;
USE WORK.ALL;

```

```

ENTITY ehxpll_top IS

    PORT (
        CLKI           :IN    std_logic;
        CLKFB          :IN    std_logic;
        RST             :IN    std_logic;
        DDAMODE        :IN    std_logic;
        DDAIZR         :IN    std_logic;
        DDAILAG        :IN    std_logic;
        DDAIDEL0       :IN    std_logic;
        DDAIDEL1       :IN    std_logic;
        DDAIDEL2       :IN    std_logic;
        CLKOP           :OUT   std_logic;
        CLKOS           :OUT   std_logic;
        CLKOK           :OUT   std_logic;
        LOCK           :OUT   std_logic;
        DDAOZR         :OUT   std_logic;
        DDAOLAG        :OUT   std_logic;
        DDAODEL0       :OUT   std_logic;
        DDAODEL1       :OUT   std_logic;
        DDAODEL2       :OUT   std_logic);

END ehxpll_top;

ARCHITECTURE V OF ehxpll_top IS
    COMPONENT EHXPLL
        GENERIC (
            FIN : string := "100";
            CLKI_DIV : string := "1";
            CLKOP_DIV : string := "1";
            CLKFB_DIV : string := "1";
            FDEL : string := "0";
            CLKOK_DIV : string := "2";
            PHASEADJ : string := "0";
            DUTY : string := "4";
            DELAY_CNTL : string := "STATIC";
            FB_MODE : string := "CLOCKTREE";
            WAKE_ON_LOCK : string := "OFF");
        PORT (
            CLKI           :IN    std_logic;
            CLKFB          :IN    std_logic;
            RST             :IN    std_logic;
            DDAMODE        :IN    std_logic;
            DDAIZR         :IN    std_logic;
            DDAILAG        :IN    std_logic;
            DDAIDEL0       :IN    std_logic;
            DDAIDEL1       :IN    std_logic;
            DDAIDEL2       :IN    std_logic;
            CLKOP           :OUT   std_logic;
            CLKOS           :OUT   std_logic;
            CLKOK           :OUT   std_logic;
            LOCK           :OUT   std_logic;
            DDAOZR         :OUT   std_logic;
            DDAOLAG        :OUT   std_logic;
            DDAODEL0       :OUT   std_logic;
            DDAODEL1       :OUT   std_logic;
            DDAODEL2       :OUT   std_logic);
    END COMPONENT;
END COMPONENT;

```

```
BEGIN
  PllInst0: EHXPLLB
    GENERIC MAP (
      FIN => "100",
      CLKI_DIV => "1",
      CLKOP_DIV => "1",
      CLKFB_DIV => "1",
      FDEL => "0",
      WAKE_ON_LOCK => "OFF",
      FB_MODE => "CLOCKTREE";
      CLKOK_DIV => "2",
      PHASEADJ => "0",
      DUTY => "4",
      DELAY_CNTL => "STATIC")
    PORT MAP (
      DDAODEL2          => DDAODEL2,
      DDAODEL1          => DDAODEL1,
      DDAODEL0          => DDAODEL0,
      DDAOLAG           => DDAOLAG,
      DDAOZR            => DDAOZR,
      LOCK              => LOCK,
      CLKOK             => CLKOK,
      CLKOS             => CLKOS,
      CLKOP             => CLKOP,
      DDAIDEL2          => DDAIDEL2,
      DDAIDEL1          => DDAIDEL1,
      DDAIDEL0          => DDAIDEL0,
      DDAILAG           => DDAILAG,
      DDAIZR            => DDAIZR,
      DDAMODE           => DDAMODE,
      RST               => RST,
      CLKFB             => CLKFB,
      CLKI              => CLKI);
END V;
```

## Appendix B. A Complete Project Example with Test Bench for Modelsim in VHDL

```

-- VHDL Test Bench --
LIBRARY ieee;
LIBRARY generics;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
USE generics.components.ALL;

ENTITY plltest_tb IS
END plltest_tb;

ARCHITECTURE behavior OF plltest_tb IS

    COMPONENT plltest
    PORT(
        sysrst      : in std_logic;
        pllrst      : in std_logic;
        sysclk0     : in std_logic;
        sysclk1     : in std_logic;
        sysclk2     : in std_logic;
        clkssel     : in std_logic;
        clklock     : out std_logic ;
        clkout_dcs  : out std_logic;
        clkout      : out std_logic;
        pllclkos    : out std_logic;
        pllclkok    : out std_logic;
        dcsreg      : out std_logic_vector(15 downto 0);
        pllreg      : out std_logic_vector(15 downto 0)
    );
    END COMPONENT;

    signal pllrst      : std_logic := '1';
    signal sysrst      : std_logic := '0';
    signal sysclk0     : std_logic := '0';
    signal sysclk1     : std_logic := '0';
    signal sysclk2     : std_logic := '0';
    signal clkssel     : std_logic := '0';
    signal clklock     : std_logic;
    signal clkout_dcs  : std_logic;
    signal clkout      : std_logic;
    signal pllclkok    : std_logic;
    signal pllclkos    : std_logic;
    signal dcscnt      : std_logic_vector(15 downto 0);
    signal pllcnt      : std_logic_vector(15 downto 0);

    constant clk_cyc0 : time := 40 ns;
    constant clk_cyc1 : time := 50 ns;

BEGIN
    sysclk0 <= not sysclk0 after clk_cyc0/2;
    sysclk1 <= not sysclk1 after clk_cyc1/2;
    sysclk2 <= not sysclk2 after clk_cyc0/2;

```

```
    uut: plltest PORT MAP
    (
        pllrst      => pllrst,
        sysrst      => sysrst,
        sysclk0     => sysclk0,
        sysclk1     => sysclk1,
        sysclk2     => sysclk2,
        clkssel     => clkssel,
        clklock     => clklock,
        clkout_dcs  => clkout_dcs,
        clkout      => clkout,
        pllclkos    => pllclkos,
        pllclkok    => pllclkok,
        dcsreg      => dcsnt,
        pllreg      => pllcnt
    );

-- *** Test Bench - User Defined Section ***
tb : PROCESS
BEGIN
    wait for 10 ns;
    sysrst <= '1';
    pllrst <= '0';

    wait for 500 ns;
    clkssel <= '1';
    wait; -- will wait forever
END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

```
-- Top Module --

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
--synthesis translate_off
library ec;
use ec.all;
--synthesis translate_on

entity plltest is
    port (
        sysrst :    in std_logic;
        pllrst  :    in std_logic;
        sysclk0 :    in std_logic;
        sysclk1 :    in std_logic;
        sysclk2 :    in std_logic;
        clkssel :    in std_logic;
        clklock :    out std_logic;
```

```

        clkout_dcs: out std_logic;
        clkout:     out std_logic;
        pllclkos:  out std_logic;
        pllclkok:  out std_logic;
        dcsreg:    out std_logic_vector(15 downto 0);
        pllreg:    out std_logic_vector(15 downto 0)
    );
end;
```

architecture behave of plltest is

```

    COMPONENT DCS
    GENERIC
    (
        DCSMODE : string := "HIGH_LOW"
    );

    PORT (
        CLK0  :IN    std_logic;
        CLK1  :IN    std_logic;
        SEL   :IN    std_logic;
        DCSOUT:OUT  std_logic
    );
END COMPONENT;
```

component ehxpll42 IS

```

    GENERIC
    (
        FIN                : string := "50.0000";
        CLKI_DIV           : string := "1";
        CLKOP_DIV          : string := "2";
        CLKFB_DIV          : string := "5";
        FDEL               : string := "3";
        CLKOK_DIV          : string := "2";
        PHASEADJ           : string := "45";
        DUTY               : string := "3";
        DELAY_CNTL         : string := "STATIC";
        FB_MODE            : string := "CLOCKTREE";
        WAKE_ON_LOCK       : string := "OFF");

    PORT
    (
        CLKI      :IN    std_logic;
        CLKFB     :IN    std_logic;
        RST       :IN    std_logic;
        DDAMODE   :IN    std_logic;
        DDAIZR    :IN    std_logic;
        DDAILAG   :IN    std_logic;
        DDAIDEL0  :IN    std_logic;
        DDAIDEL1  :IN    std_logic;
        DDAIDEL2  :IN    std_logic;
        CLKOP     :OUT   std_logic;
        CLKOS     :OUT   std_logic;
        CLKOK     :OUT   std_logic;
    );
```

```

        LOCK      :OUT   std_logic;
        DDAOZR    :OUT   std_logic;
        DDAOLAG   :OUT   std_logic;
        DDAODEL0  :OUT   std_logic;
        DDAODEL1  :OUT   std_logic;
        DDAODEL2  :OUT   std_logic
    );

END component;

    signal dcsclk      : std_logic;
    signal pllclk      : std_logic;
    signal dcsreg_int  : std_logic_vector(15 downto 0);
    signal pllreg_int  : std_logic_vector(15 downto 0);

begin

DCSInst0: DCS
    GENERIC MAP
        (
            DCSMODE      => "POS"
        )
    PORT MAP
        (
            SEL          => clkssel,
            CLK0         => sysclk0,
            CLK1         => sysclk1,
            DCSOUT       => dcsclk
        );

clkout_dcs <= dcsclk;

PLLInst0: ehxpll42
    GENERIC MAP
        (
            FIN          => "50",
            CLKI_DIV     => "1",
            CLKOP_DIV    => "2",
            CLKFB_DIV    => "5",
            FDEL         => "0",
            WAKE_ON_LOCK => "OFF",
            FB_MODE      => "CLOCKTREE";
            CLKOK_DIV    => "2",
            PHASEADJ     => "0",
            DUTY         => "3",
            DELAY_CNTL   => "STATIC"
        )

    port map
        (
            RST          => pllrst,
            CLKI         => dcsclk,
            CLKFB        => pllclk,
            CLKOP        => pllclk,

```

---

```

        LOCK          => clklock,
        CLKOS         => pllclkos,
        CLKOK         => pllclkok,
        DDAMODE       => '0',
        DDAIZR        => '0',
        DDAILAG       => '0',
        DDAIDEL0      => '0',
        DDAIDEL1      => '0',
        DDAIDEL2      => '0',
        DDAOZR        => OPEN,
        DDAOLAG       => OPEN,
        DDAODEL0      => OPEN,
        DDAODEL1      => OPEN,
        DDAODEL2      => OPEN
    );

clkout <= pllclk;

dcsinst1: process(dcsclk, sysrst)
begin
    if (sysrst = '0') then
        dcsreg_int <= (others => '0');
    elsif rising_edge(dcsclk) then
        dcsreg_int <= dcsreg_int + 1;
    end if;
end process;
dcsreg <= dcsreg_int;

pllinst1: process pllclk, sysrst)
begin
    if (sysrst = '0') then
        pllreg_int <= (others => '0');
    elsif rising_edge pllclk) then
        pllreg_int <= pllreg_int + 1;
    end if;
end process;
pllreg <= pllreg_int;

end behave;

-- EHXPLLB Module Generated by Module Manager --

USE STD.TEXTIO.ALL;
Library IEEE, EC;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_arith.ALL;
USE EC.COMPONENTS.ALL;
USE WORK.ALL;
ENTITY ehxpll42 IS

    GENERIC
    (
        FIN                : string := "50.0000";
        CLKI_DIV           : string := "1";

```

---

```

        CLKOP_DIV      : string := "2";
        CLKFB_DIV      : string := "5";
        FDEL           : string := "3";
        CLKOK_DIV      : string := "2";
        PHASEADJ       : string := "45";
        DUTY           : string := "3";
        DELAY_CNTRL    : string := "STATIC";
        FB_MODE        : string := "CLOCKTREE";
        WAKE_ON_LOCK   : string := "OFF"
    );

PORT
    (
        CLKI           :IN      std_logic;
        CLKFB          :IN      std_logic;
        RST             :IN      std_logic;
        DDAMODE        :IN      std_logic;
        DDAIZR         :IN      std_logic;
        DDAILAG        :IN      std_logic;
        DDAIDEL0       :IN      std_logic;
        DDAIDEL1       :IN      std_logic;
        DDAIDEL2       :IN      std_logic;
        CLKOP          :OUT     std_logic;
        CLKOS          :OUT     std_logic;
        CLKOK          :OUT     std_logic;
        LOCK           :OUT     std_logic;
        DDAOZR         :OUT     std_logic;
        DDAOLAG        :OUT     std_logic;
        DDAODEL0       :OUT     std_logic;
        DDAODEL1       :OUT     std_logic;
        DDAODEL2       :OUT     std_logic
    );

END ehxpll42;

ARCHITECTURE V OF ehxpll42 IS
    COMPONENT EHXPLLB
        GENERIC
            (
                FIN           : string := "50.0000";
                CLKI_DIV     : string := "1";
                CLKOP_DIV     : string := "2";
                CLKFB_DIV     : string := "5";
                FDEL          : string := "3";
                CLKOK_DIV     : string := "2";
                PHASEADJ      : string := "45";
                DUTY          : string := "3";
                DELAY_CNTRL   : string := "STATIC";
                FB_MODE       : string := "CLOCKTREE";
                WAKE_ON_LOCK  : string := "OFF"
            );
        PORT
            (
                CLKI          :IN      std_logic;

```

```

        CLKFB      :IN      std_logic;
        DDAMODE    :IN      std_logic;
        DDAIZR     :IN      std_logic;
        DDAILAG    :IN      std_logic;
        DDAIDEL0   :IN      std_logic;
        DDAIDEL1   :IN      std_logic;
        DDAIDEL2   :IN      std_logic;
        CLKOP      :OUT     std_logic;
        CLKOS      :OUT     std_logic;
        CLKOK      :OUT     std_logic;
        LOCK       :OUT     std_logic;
        DDAOZR     :OUT     std_logic;
        DDAOLAG    :OUT     std_logic;
        DDAODEL0   :OUT     std_logic;
        DDAODEL1   :OUT     std_logic;
        DDAODEL2   :OUT     std_logic
    );
END COMPONENT;

BEGIN
    PllInst0: EHXPLLB
        GENERIC MAP
            (
                FIN              => "50.0000",
                CLKI_DIV         => "1",
                CLKOP_DIV        => "2",
                CLKFB_DIV        => "5",
                FDEL              => "3",
                WAKE_ON_LOCK     => "OFF",
                FB_MODE          => "CLOCKTREE";
                CLKOK_DIV        => "2",
                PHASEADJ         => "45",
                DUTY              => "3",
                DELAY_CNTL       => "STATIC"
            )
        PORT MAP
            (
                DDAODEL2        => DDAODEL2,
                DDAODEL1        => DDAODEL1,
                DDAODEL0        => DDAODEL0,
                DDAOLAG         => DDAOLAG,
                DDAOZR          => DDAOZR,
                LOCK            => LOCK,
                CLKOK           => CLKOK,
                CLKOS           => CLKOS,
                CLKOP           => CLKOP,
                DDAIDEL2        => DDAIDEL2,
                DDAIDEL1        => DDAIDEL1,
                DDAIDEL0        => DDAIDEL0,
                DDAILAG         => DDAILAG,
                DDAIZR          => DDAIZR,
                DDAMODE         => DDAMODE,
                RST             => RST,
                CLKFB           => CLKFB,

```

```
        CLKI      => CLKI  
    );  
  
END V;
```