LatticeECP2/M sysCLOCK PLL/DLL Design and Usage Guide

June 2013 Technical Note TN1103

Introduction

This user's guide describes the clock resources available in the LatticeECP2™ and LatticeECP2M™ device architectures. Details are provided for primary clocks, secondary clocks and edge clocks, as well as clock elements such as PLLs, DLLs, Clock Dividers and more.

The number of PLLs and DLLs for each package can be found in Tables 10-1 and 10-2.

Table 10-1. Number of PLLs and DLLs: LatticeECP2 Family

Device	Description	ECP2-6	ECP2-12	ECP2-20	ECP2-35	ECP2-50	ECP2-70
Number of SPLLs	Standard PLL (Subset of GPLL)	0	0	0	0	2	4
Number of GPLLs	General Purpose PLL	2	2	2	2	2	2
Number of DLLs	General Purpose DLL	2	2	2	2	2	2
Number of DQSDLLs	DLL for DDR Applications	2	2	2	2	2	2

Table 10-2. Number of PLLs, DLLs and SERDES: LatticeECP2M Family

Device	Description	ECP2M-20	ECP2M-35	ECP2M-50	ECP2M-70	ECP2M-100
Number of SPLLs	Standard PLL (Subset of GPLL)	6	6	6	6	6
Number of GPLLs	General Purpose PLL	2	2	2	2	2
Number of DLLs	General Purpose DLL	2	2	2	2	2
Number of DQSDLLs	DLL for DDR Applications	2	2	2	2	2
SERDES	4-Channel Quad SERDES	1	1	2	4	4

Clock/Control Distribution Network

The LatticeECP2/M family provides global clock distribution in the form of eight quadrant-based primary clocks and flexible secondary clocks. The devices also provide two edge clocks on each edge of the device. Other clock sources include clock input pins, internal nodes, PLLs, DLLs, Slave Delay Lines and Clock Dividers.

LatticeECP2/M Top Level View

Figure 10-1 shows the primary clocking structure of the LatticeECP2-50 device.

Figure 10-1. LatticeECP2-50 Clocking Structure

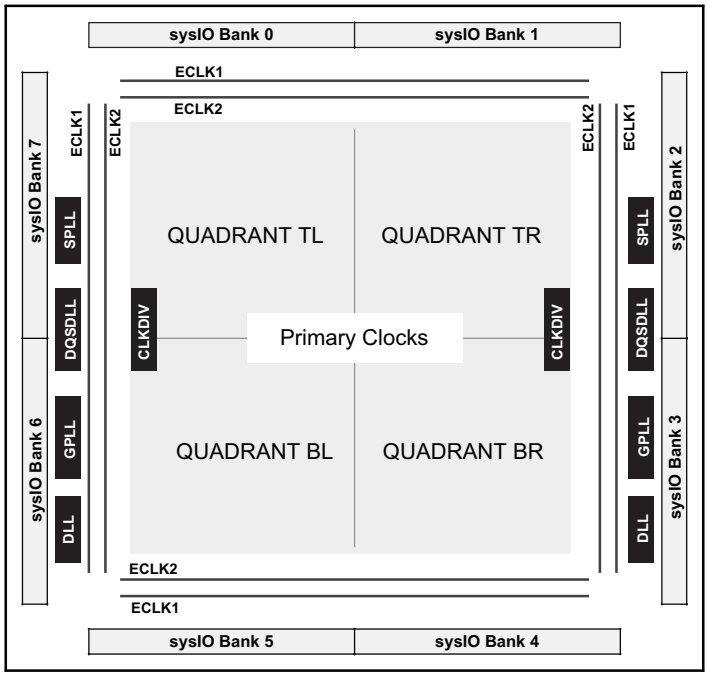
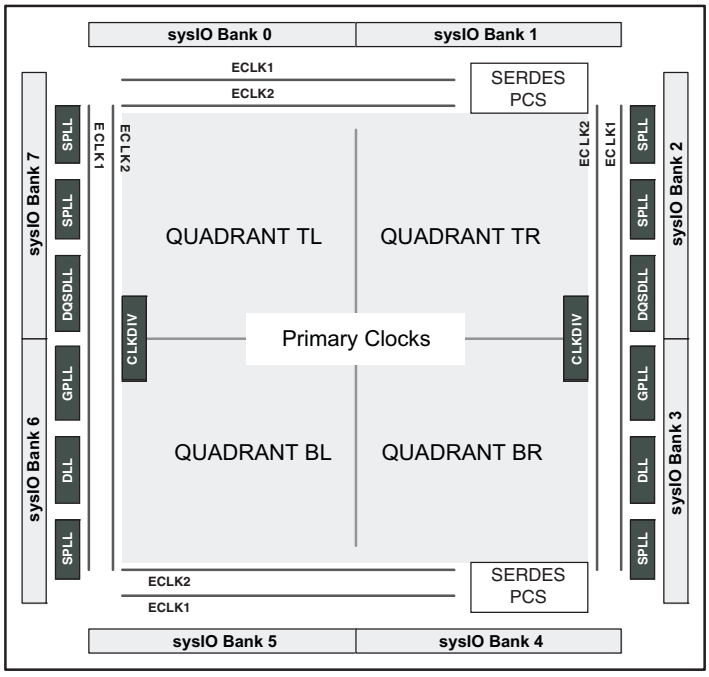


Figure 8-2 illustrates the primary clocking structure of the LatticeECP2M-50 device. The figure shows two SERDES blocks. The edge clocks on the top and bottom sides stop when they reach the SERDES block boundary. Other members of the LatticeECP2M family have a similar structure, except for the number of SERDES blocks.

Figure 10-2. LatticeECP2M-50 Clocking Structure



Primary Clocks

Each quadrant receives up to eight primary clocks. Two of these clocks provide the dynamic clock selection (DCS) feature. The six primary clocks without DCS can be specified in the Spreadsheet View in the ispLEVER Design Planner (or **Tools > Spreadsheet View** in the Lattice Diamond™ design software) as 'Primary Pure' and the two DCS clocks as 'Primary-DCS'.

The sources of the primary clocks are:

- PLL outputs
- DLL outputs
- CLKDIV outputs
- Dedicated clock pins
- Internal nodes
- SERDES TX_H_CLK (LatticeECP2M only)

Secondary Clocks

The LatticeECP2/M secondary clocks are a flexible region-based clocking resource. Each region can have four independent clock inputs. As a regional resource, it can cross the primary clock quadrant boundaries.

There are eight secondary clock muxes per region. Each mux has inputs from four different sources. Three of these are from internal nodes. The fourth input comes from a primary clock pin. The input sources are not necessarily located in the same region as the Mux. This structure enables global usage of secondary clocks.

The sources of secondary clocks are:

- Dedicated clock pins on right and left sides of device (PCLKT2, PCLKT3, PCLKT6, PCLKT7)
- Internal nodes

Edge Clocks

The LatticeECP2/M has two edge clocks per side. These clocks, which have low injection times and skew, are used to clock I/O registers. The edge clock (ECLK) resources are designed for high speed I/O interfaces with high fanout capability. Refer to Appendix B for detailed connectivity information.

The sources of the edge clocks are:

- Left and Right Edge Clocks
  - Dedicated clock pins
  - PLL outputs
  - DLL outputs
  - Internal nodes
- Top and Bottom Edge Clocks
  - Dedicated clock pins
  - Internal nodes

ECLK can directly drive the secondary clock resources and general routing resources. This means that an ECLK source clock can also route to the Primary Clock Net through general routing at the same time.

Figure 10-3 describes the secondary clock and edge clock structure.

Figure 10-3. LatticeECP2-50 Secondary Clocks and Edge Clocks

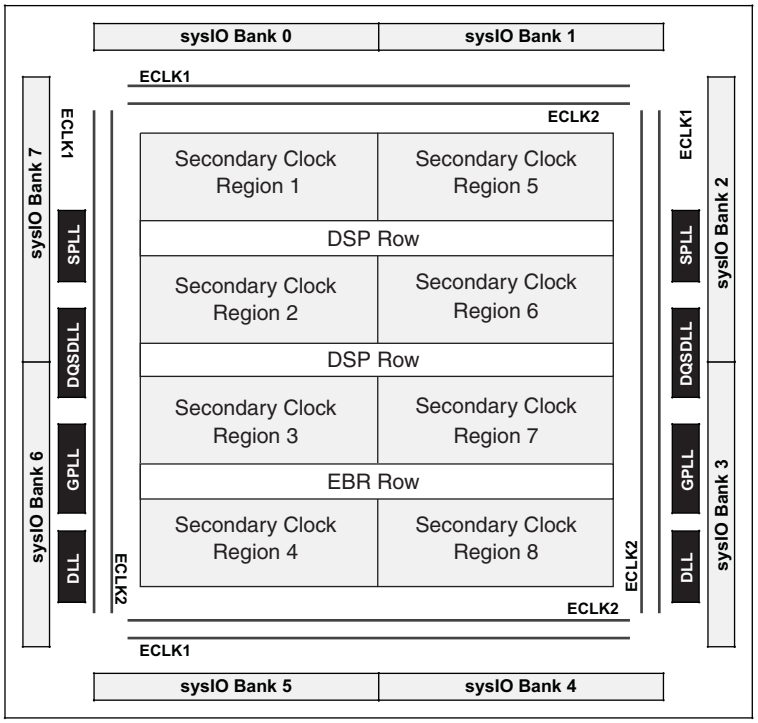
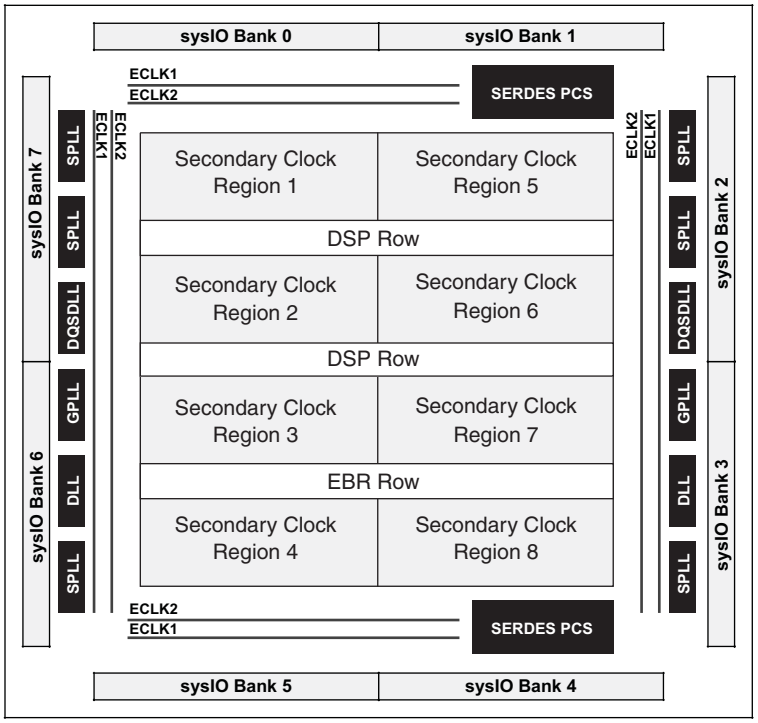


Figure 10-4. LatticeECP2M-50 Secondary Clocks and Edge Clocks



Note on Primary Clocks

The CLKOP must be used as the feedback source to optimize the PLL performance.

Most designers use PLL for clock tree injection removal mode and the CLKOP should be assigned to the Primary Clock. This is done automatically by the software unless the user specifies otherwise.

CLKOP can route to CLK0 to CLK5 only and CLKOS/CLKOK can route to all Primary Clocks (CLK0 to CLK7).

When CLK6 or CLK7 is used as a Primary Clock and there is only one clock input to the DCS, the DCS is assigned as a buffer mode by the software. See the DCS section of this document for further information.

Specifying Clocks in the Design Tools

If desired, designers can specify the clock resources, primary, secondary or edge to be used to distribute a given clock source. Figure 10-4 illustrates how this can be done in the Spreadsheet View in the ispLEVER Design Planner (or **Tools > Spreadsheet View** in Diamond). Alternatively the Preference file can be used, as discussed in Appendix C.

Primary-Pure and Primary-DCS

Primary Clock Net can be assigned to either Primary-Pure (CLK0 to CLK5) or Primary-DCS (CLK6 and CLK7).

Global Primary Clock and Quadrant Primary Clock

Global Primary Clock

If a primary clock is not assigned as a quadrant clock, the software assumes it is a Global Clock.

There are six Global Primary/Pure Clocks and two Global Primary/DCS Clocks available.

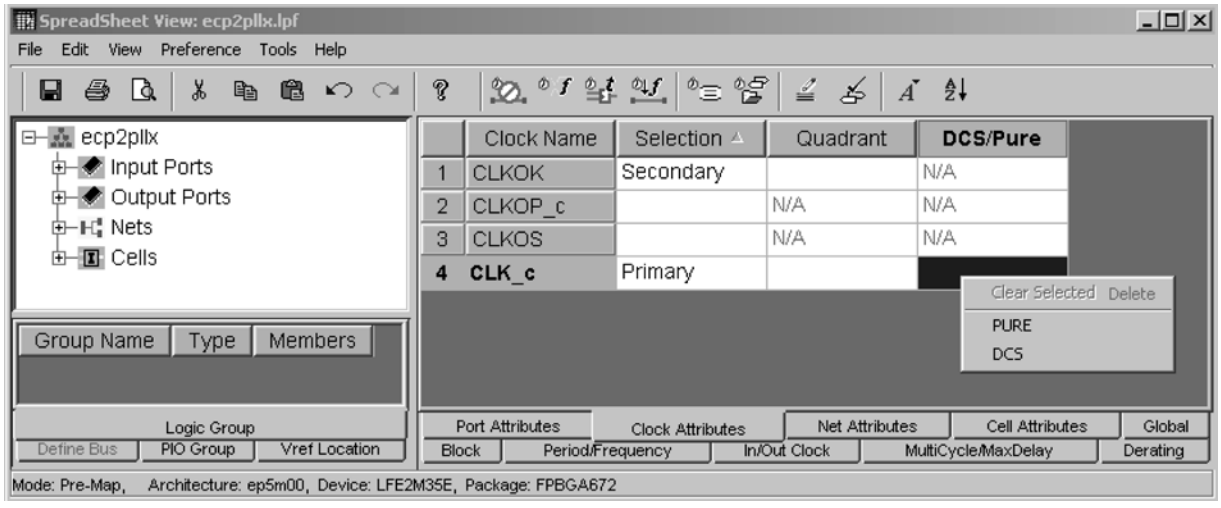
Quadrant Primary Clock

Any Primary Clock may be assigned to a Quadrant Clock. The clock may be assigned to a single quadrant or to two adjacent quadrants (not diagonally adjacent).

When a quadrant clock net is used, the user must ensure that the registers each clock drives can be assigned in that quadrant without any routing issues.

In the Quadrant Primary Clocking scheme, the maximum number of Primary Clocks is 32, as long as all the Primary Clock sources are available.

Figure 10-5. Design Planner Spreadsheet View (see Appendix D Figure 10-39 for Diamond Equivalent)



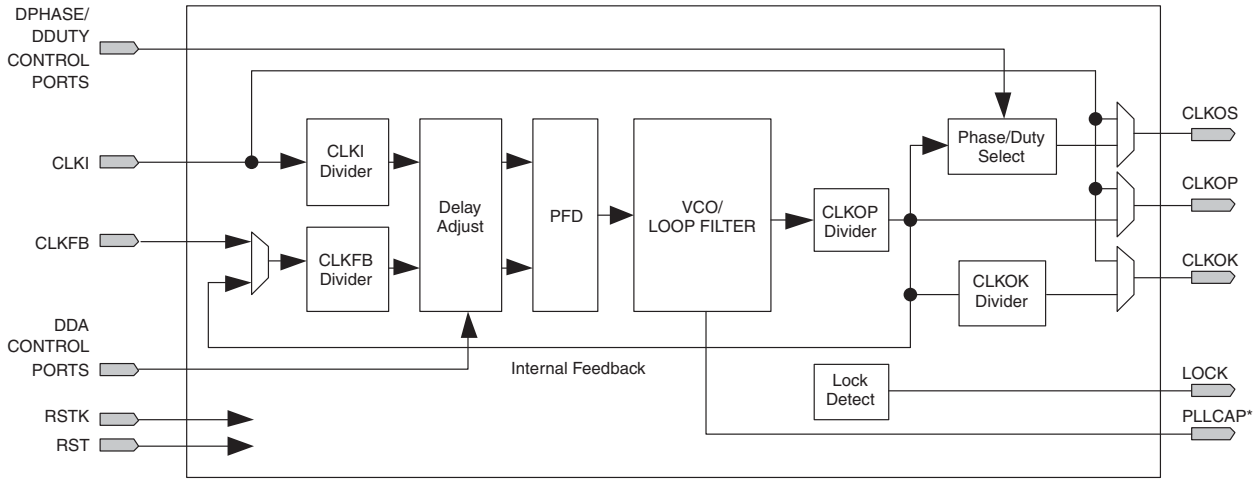
Note on Edge Clocks

Refer to Appendix A for detailed clock network diagrams.

sysCLOCK PLL

The LatticeECP2/M PLL provides features such as clock injection delay removal, frequency synthesis, phase/duty cycle adjustment, and dynamic delay adjustment. Figure 10-6 shows the block diagram of the PLL.

Figure 10-6. PLL Block Diagram



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. 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. 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 130ps (nominal) for a total of 1.04ns, lagging or leading. The time delay setting has a tolerance. See the 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 640MHz to 1280MHz range to minimize jitter. The CLKOP Divider values are the same whether or not CLKOS is used.

CLKOK Divider

The CLKOK divider acts as a source for the global clock nets. It divides the CLKOP signal of the PLL by the value of the divider to produce a lower frequency clock.

Phase Adjustment and Duty Cycle Select

Users can program CLKOS with Phase and Duty Cycle options. Phase adjustment can be done in 22.5° steps. The Duty Cycle resolution is 1/16th of a period. However, 1/16th and 15/16th duty cycle options are not supported to avoid minimum pulse violation.

Dynamic Phase Adjustment (DPHASE) and Dynamic Duty Cycle (DDUTY) Select

With LatticeECP2/M device families, users can control the Phase Adjustment and Duty Cycle Select in dynamic mode. When this mode is selected, both the Phase Adjustment and Duty Cycle Select must be in Dynamic mode. If only one of the features is to be used in Dynamic mode, the other control inputs can be set with the fixed logic levels desired.

External Capacitor

An optional external capacitor can be used with PLLs to accommodate low frequency input clocks. See the Optional External Capacitor section of this document for further information.

PLL Inputs and Outputs

CLKI Input

The CLKI signal is the reference clock for the PLL. It must conform to the specifications in the LatticeECP2/M Family Data Sheet 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, flip-flops (including M-Dividers), and the charge pump. The M-Divider reset synchronizes the M-Divider output to the input clock. 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 10-7 shows the timing diagram of the RST Input. RST is active high.

The RESET signal is optional.

Figure 10-7. RST Input Timing Diagram

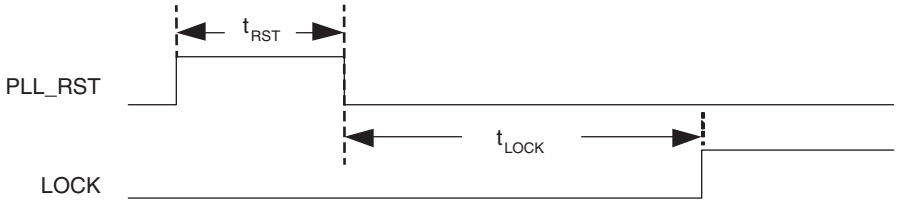
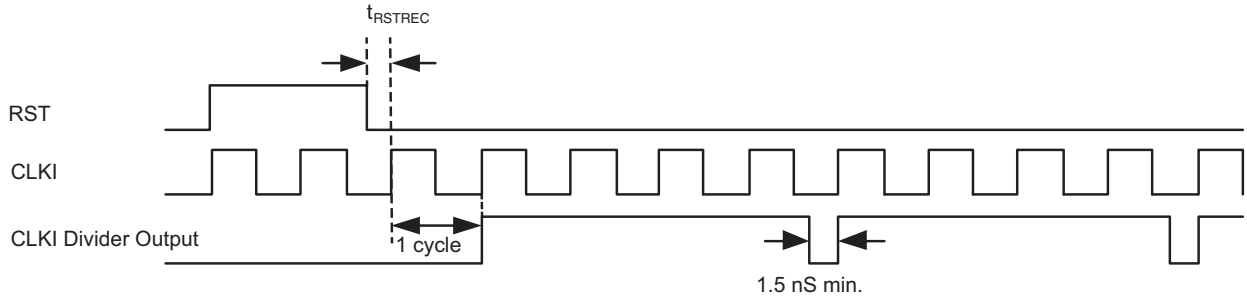


Figure 10-8 shows the timing relationship between RST and the CLKI Divider Output.

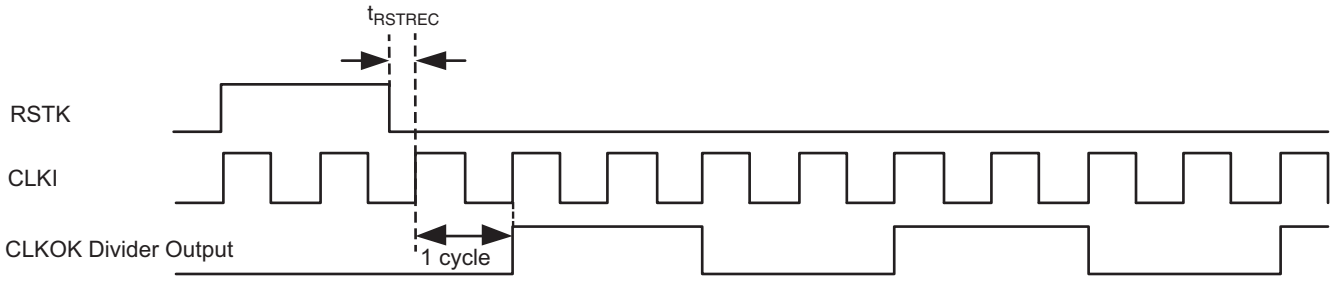
Figure 10-8. RST Input and CLKI Divider Output Timing Diagram (Example: CLKI_DIV = 4)



RSTK Input
RSTK is the reset input for the K-Divider. The K-Divider reset is used to synchronize the K-Divider output clock to the input clock. The LatticeECP2/M has an optional gearbox in the I/O cell for both outputs and inputs. The K-Divider reset is useful for the gearbox implementation. RSTK is active high.

Figure 10-9 shows the timing relationship between RSTK and CLKOK (example: CLKOK_DIV = 4)

Figure 10-9. RSTK Input and CLKOK Divider Output Timing Diagram (Example: CLKOK_DIV = 4)



CLKFB Input
The feedback signal to the PLL, which is fed through the feedback divider, can be derived from the Primary Clock net (CLKOP), a preferred pin, directly from the CLKOP divider (internal feedback) or from general routing. 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 and an edge clock. This clock signal is available at the CLK_OUT pin.

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 and an edge clock. The CLKOS is used when phase shift and/or duty cycle adjustment is desired. The programmable phase shift allows for different phases in increments of 22.5°. The duty select feature provides duty selection in 1/16th of the clock period. This feature is also supported in Dynamic Control Mode.

CLKOK Output with Lower Frequency
The CLKOK is used when a lower frequency is desired. This signal is available for selection as a primary clock.

Dynamic Delay Control/Dynamic Phase Adjustment/Dynamic Duty Cycle
Detailed information about these features are described later in this document. The I/O ports for these features are illustrated in Table 10-3.

Table 10-3. Dynamic Delay Adjust and Dynamic Phase and Duty Cycle Adjust Ports

Parameter	I/O	Description
DDAMODE	I	DDA (Dynamic Delay Adjust) Mode. 1": Pin control (dynamic), “0”: Fuse Control (static)
DDAIZR	I	DDA Delay Zero. “1”: delay = 0, “0”: delay = on
DDAILAG	I	DDA Lag/Lead. “1”: Lead, “0”: Lag
DDAIDEL[2:0}	I	DDA Delay Step value
DPAMODE	I	DPA (Dynamic Phase Adjust/Duty Cycle Select) mode. 1": Pin Pin control (dynamic), “0”: Fuse Control (static)
DPHASE[3:0]	I	DPA Phase Adjust inputs
DDUTY[3:0]	I	DPA Duty Cycle Select inputs

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. However, when the input clock completely stops, the LOCK output will remain in its last state, since it is internally registered by this clock. It is recommended to assert PLL RST to re-synchronize the PLL to the reference clock. The LOCK signal is available to the FPGA routing to implement generation of RST. Simulation models take several reference clock cycles from RST release to LOCK high.

PLLCAP
This port is not included in the software module. Instead, it is hard-wired to the PLLCAP pin of the device. See the Optional External Capacitor section of this document for further information.

PLL Attributes

The PLL utilizes several attributes that allow the configuration of the PLL through source constraints and preference files. 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. This is determined by the input frequency, the dividers and the PLL specifications.

Note: Unlike PLLs in the LatticeECP™, LatticeEC™, LatticeXP™ and MacoXO™ devices, the CLKOP Divider values are the same whether or not CLKOS is used. The CLKOP_DIV value is calculated to maximize the f_{VCO} within the specified range based on FIN and CLKOP_FREQ in conjunction with CLKI_DIV and CLKFB_DIV values. These value settings are designed so that the output clock duty cycle is as close to 50% as possible.

FREQUENCY_PIN_CLKI, FREQUENCY_PIN_CLKOP, FREQUENCY_PIN_CLKOK
These input and output clock frequencies determine the divider values.

CLKOP Frequency Tolerance
When the desired output frequency is not achievable, the frequency tolerance of the clock output may be entered.

PHASEADJ (Phase Shift Adjust)
The PHASEADJ attribute is used to select Phase Shift for the CLKOS output. The phase adjustment is programmable in 22.5° increments.

DUTY (Duty Cycle)
The DUTY attribute is used to select the Duty Cycle for CLKOS output. The Duty Cycle is programmable at 1/16th of the period increment. Steps 2 to 14 are supported. 1/16th and 15/16th duty cycles are not supported to avoid the minimum pulse width violation.

FB_MODE
There are three sources of feedback signals that can drive the CLKFB Divider: Internal, CLKOP (Clock Tree) and User Clock. CLKOP (Clock Tree) feedback is used by default. Internal feedback takes the CLKOP output at the CLKOP Divider output (CLKINTFB) before the Clock Tree to minimize the feedback path delay. User Clock feedback is driven from the dedicated pin, clock pin or user specified internal logic.

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 the DDAMODE pin. If the attribute is set to “STATIC”, Dynamic Delay inputs are ignored in this mode.

PHASE/DUTY_CNTL
This attribute is designed to select the Phase Adjustment/Duty Cycle Select mode. If the attribute is set to "DYNAMIC" the Phase Adjustment/Duty Cycle Select control switches between dynamic and static, depending upon the input logic of the DPAMODE pin. If the attribute is set to "STATIC", Dynamic Phase Adjustment/Duty Cycle Select inputs are ignored in this mode.

CLKOS/CLKOK Select
Users select these output clocks only when they are used in the design.

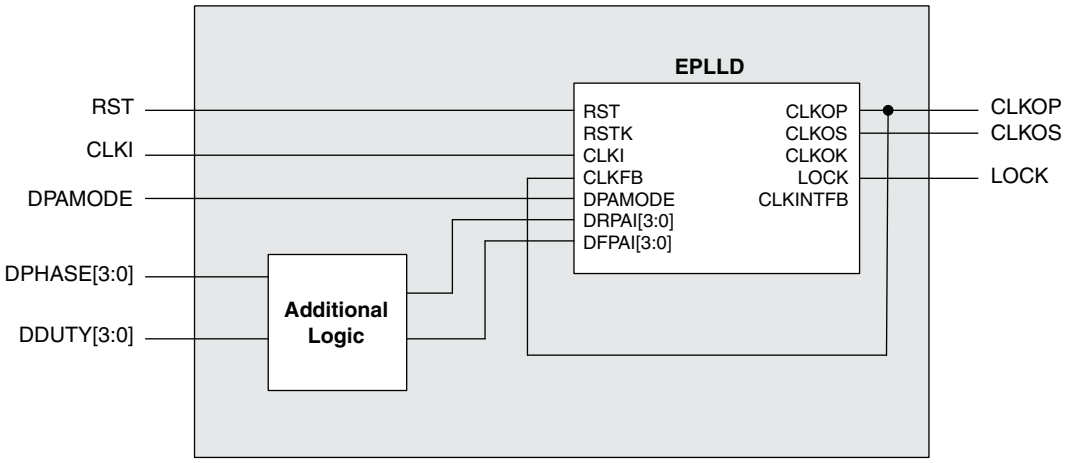
CLKOP/CLKOS/CLKOK BYPASS
These bypasses are enabled if set. The CLKI is routed directly to the corresponding output clock.

RESET/RSTK Select
Users select these reset signals only when they are used in the design.

LatticeECP2/M PLL Modules

When the user creates a PLL module using IPexpress, the module will consist of a wrapper around the PLL library element and any additional logic required for the module. Figure 10-10 shows a diagram of a typical PLL module. The module port names can be different than the library element is some cases. The user will see the module port names in the IPexpress window and also in the source code file for the generated module. These are the ports that will be connected in the user's design. IPexpress also creates an instantiation template file that shows the user how to instantiate the PLL module in their design. The user can import the *.LPC (or *.IPX for Diamond) file into their project or the generated source code file.

Figure 10-10. LatticeECP2/M Typical PLL Module Generated by IPexpress



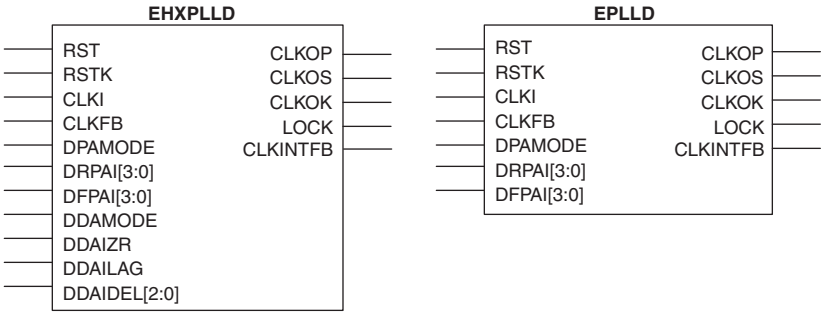
The PLL module shown in Figure 10-10 represents an example where the user has chosen to use the CLKOP and CLKOS ports, with a PLL reset signal, PLL lock signal, and dynamic phase and dynamic duty cycle. It also uses CLKOP feedback so the software will connect the CLKOP signal to the CLKFB port and use the primary clock tree to route this signal. The user would connect their signals to the CLKI, RST, DPAMODE, DPHASE[3:0], DDUTY[3:0], CLKOP, CLKOS, and LOCK signals.

LatticeECP2/M PLL Library Definitions

All LatticeECP2/M devices support two General Purpose PLLs (GPLLs) which are full-featured PLLs. In addition, some of the larger devices have two to six Standard PLLs (SPLLs) that have a subset of the GPLL functionalities.

Two PLL library elements are defined for LatticeECP2/M PLL implementation. Figure 10-11 shows the LatticeECP2/M PLL library symbols. The GPLL may be configured as either EPLLD or EHXPLLD. The SPLL can be configured as EPLLD only.

Figure 10-11. LatticeECP2/M PLL Library Symbols



Dynamic Delay Adjustment (EHXPLLD Only)

The Dynamic Delay Adjustment is controlled by the DDAMODE input. 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 Tdly = 130 ps (nominal)	DDAMODE = 0
DDAIZR	DDAILAG	DDAIDEL[2:0]		Equivalent FDEL Value
0	1	111	Lead 8 Tdly	-8
0	1	110	Lead 7 Tdly	-7
0	1	101	Lead 6 Tdly	-6
0	1	100	Lead 5 Tdly	-5
0	1	011	Lead 4 Tdly	-4
0	1	010	Lead 3 Tdly	-3
0	1	001	Lead 2 Tdly	-2
0	1	000	Lead 1 Tdly	-1
1	Don't Care	Don't Care	no delay	0
0	0	000	Lag 1 Tdly	1
0	0	001	Lag 2 Tdly	2
0	0	010	Lag 3 Tdly	3
0	0	011	Lag 4 Tdly	4
0	0	100	Lag 5 Tdly	5
0	0	101	Lag 6 Tdly	6
0	0	110	Lag 7 Tdly	7
0	0	111	Lag 8 Tdly	8

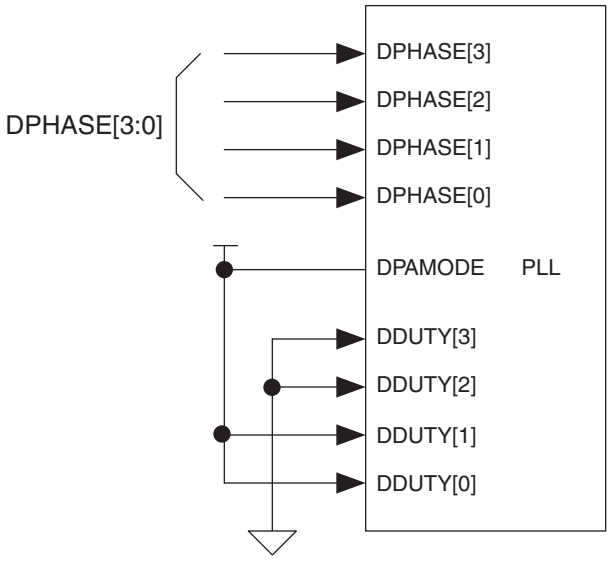
Dynamic Phase/Duty Mode

This mode sets both Dynamic Phase Adjustment and Dynamic Duty Select at the same time.

There are two modes, "Dynamic Phase and Dynamic Duty" and "Dynamic Phase and 50% Duty".

To use Dynamic Phase Adjustment with a fixed Duty Cycle, simply set the DDUTY[3:0] inputs to the desired Duty Cycle value. Figure 10-12 illustrates an example circuit. This example assumes the user-desired Duty Cycle is 3/16.

Figure 10-12. Example Dynamic Phase Adjustment Set-up with Duty Cycle Fixed to 3/16



Dynamic Phase Adjustment/Duty Cycle Select

Phase Adjustment settings are described in Table 10-5.

Table 10-5. Dynamic Phase Adjustment Settings

DPHASE[3:0]	Equivalent to PHASEADJ in Static Mode
0000	0
0001	22.5
0010	45
0011	67.5
0100	90
0101	112.5
0110	135
0111	157.5
1000	180
1001	202.5
1010	225
1011	247.5
1100	270
1101	292.5
1110	315
1111	337.5

Duty Cycle Select settings are described in Table 10-6.

Table 10-6. Dynamic Duty Cycle Select Settings

DDUTY[3:0]	Equivalent to DUTY in Static Mode (1/16 of Period)	Comment
0000	0	Not Supported
0001	1	Not Supported
0010	2	
0011	3	
0100	4	
0101	5	
0110	6	
0111	7	
1000	8	
1001	9	
1010	10	
1011	11	
1100	12	
1101	13	
1110	14	
1111	15	Not Supported

Optional External Capacitor

An optional external capacitor can be used with both the EHXPLLD and the EPLLD to change the frequency response of the on-chip loop filter. When an external capacitor is used, the frequency at the phase detector inputs (Fpd) can be as low as 2MHz, allowing the PLLs to extend the low-end of their operating ranges. Using the external capacitor will limit the high end of the PLL operating range as shown in the LatticeECP2/M Family Data Sheet. IPexpress™ checks the phase detector frequency to determine if an external capacitor is required.

The allowable ranges for the PLL parameters with and without the external capacitor are described in the LatticeECP2/M Family Data Sheet.

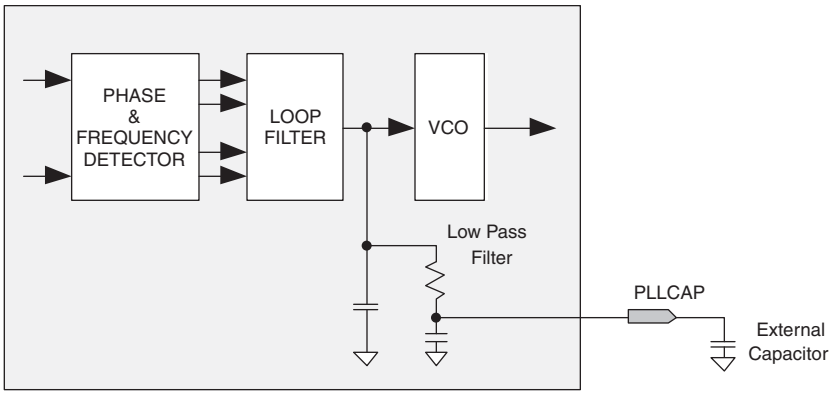
Recommended Optional External Capacitor Specifications

Value: 5.6 nF, +/- 20%
Type: Ceramic chip capacitor, NPO dielectric
Package: 1206 or smaller

Each device has two external capacitor pins, one for the left side PLLs and one for the right side PLLs. These pins are in fixed locations. They are dedicated function pins that are NOT shared with user I/Os.

When an external capacitor pin is used by a PLL on one side of the device, it cannot be used by any other PLLs on the same side of the device. This means that a maximum of two PLLs per device, one on the left side and one on the right side, can have external capacitors attached.

Figure 10-13. External Capacitor Usage

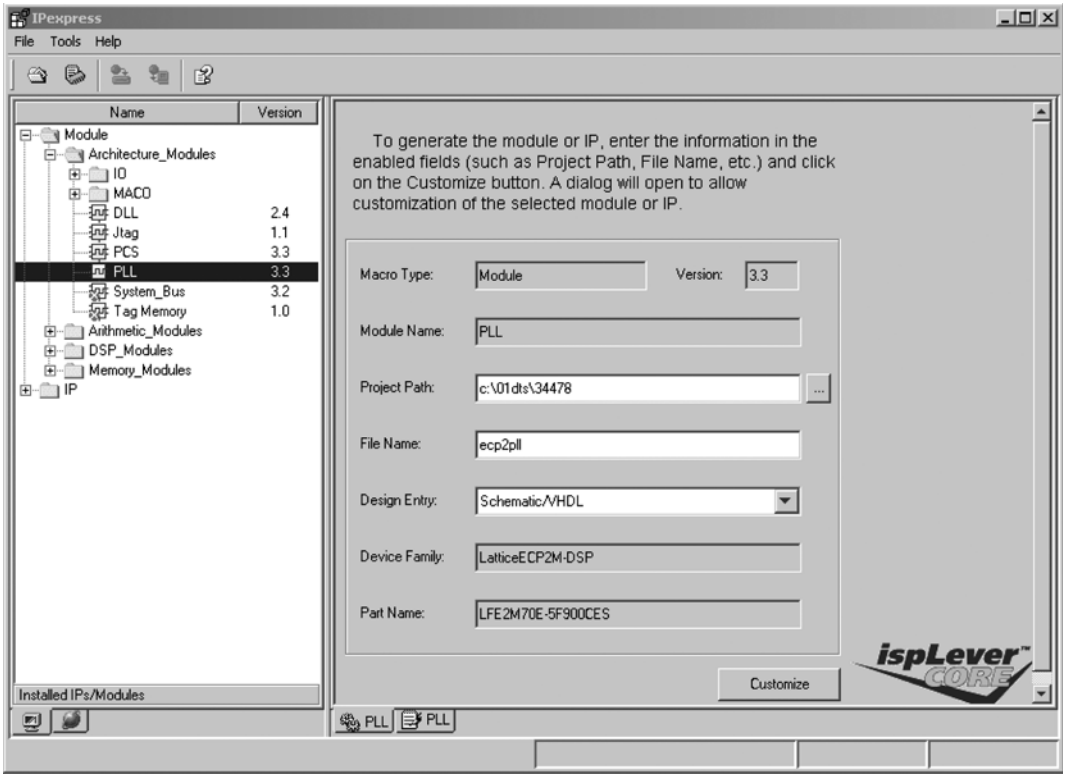


PLL Usage in IPexpress

IPexpress is used to create and configure a PLL. Designers use the graphical user interface to select parameters for the PLL. The result is an HDL model to be used in the simulation and synthesis flow.

Figure 10-14 shows the main window when PLL is selected. The only entry required in this window is the module name. Other entries are set to the project settings. These entries may be changed if desired. After entering the module name, click on **Customize** to open the **Configuration Tab** window as shown in Figure 10-15.

Figure 10-14. IPexpress Main Window (see Appendix D Figure 10-40 for Diamond Equivalent)



Configuration Tab

The Configuration Tab lists all user-accessible attributes with default values set. Upon completion, click **Generate** to generate source and constraint files. The user may choose to use the *.LPC file (or *.IPX file for Diamond projects) to load parameters.

Modes

There are two modes for configuring the PLL in the Configuration Tab: Frequency Mode and Divider Mode.

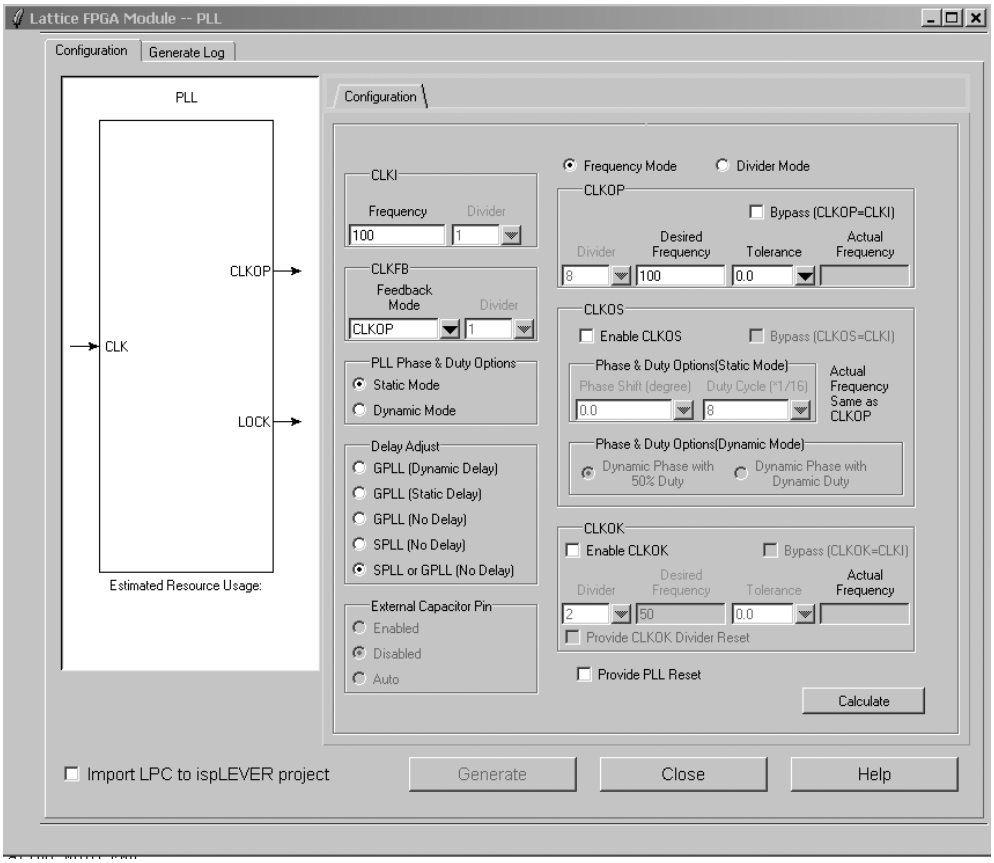
Frequency Mode

In this mode, the user enters input and output clock frequencies and the software calculates the divider settings. If the output frequency 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** button will display the divider values.

Divider Mode

In this mode, the user sets the divider settings with the input frequency. The user must choose the CLKOP Divider value in order to maximize the f_{VCO} and achieve optimum PLL performance. After setting the input frequency and divider settings, click **Calculate** to display the frequencies. Figure 10-15 shows the Configuration Tab.

Figure 10-15. LatticeECP2/M PLL Configuration Tab



Note: In the External Capacitor Pin, the grayed out text will automatically indicate the requirement for the external loop capacitor based upon the PLL settings. This is used to alert the user that the external loop capacitor may be required. The Auto setting indicates that the software will determine if the external loop capacitor is required after the PLL is placed into an SPLL or GPLL by the Place and Route (PAR) step. Other grayed-out sections of this dialog box turn on as their sections are enabled.

Table 10-7 describes all user parameters in the IPexpress GUI.

Table 10-7. User Parameters in the Configuration GUI

User Parameters		Description	Range	Default
Frequency Mode		Desired input/output frequency	ON/OFF	ON
Divider Mode		Desired input frequency and divider settings	ON/OFF	OFF
CLKI	Frequency	Without external capacitor	25 (33[1]) MHz to 420 MHz	100 MHz
		With external capacitor	2 MHz to 420 MHz[2]	—
	Divider	Without external capacitor	1 to 16(12[1])	1
		With external capacitor	1 to 64	—
CLKFB	Feedback Mode	Feedback Mode	Internal, CLKOP, User clock	CLKOP
	Divider	Without external capacitor	1 to 16 (12[1])	1
		With external capacitor	1 to 10	-
CLKOP	Bypass	Bypass PLL: CLKOP = CLKI	ON/OFF	OFF
	Desired Frequency	Without external capacitor	25(331) MHz to 420 MHz	100 MHz
		With external capacitor	5 MHz to 50 MHz[3]	—
	Divider	CLKOP Divider Setting (Divider Mode)	2,4,8,16,32,48, 64,80,96,112,128	8
	Tolerance	CLKOP tolerance users can tolerate	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0	0.0
	Actual Frequency	Actual frequency achievable, Read only	—	—
CLKOS	Enable	Enable CLKOS output clock	ON/OFF	OFF
	Bypass	Bypass PLL: CLKOS = CLKI	ON/OFF	OFF
	Phase - Static	CLKOS Static Phase Shift	0°, 22.5°, 45°, ... , 337.5°	—
	Duty - Static	CLKOS Static Duty Cycle Select	2 to 14	8
	Dynamic Phase with 50% Duty	Dynamic Phase and 50% Duty Cycle	ON/OFF	ON
	Dynamic Phase with Dynamic Duty	Dynamic Phase and Dynamic Duty Cycle	ON/OFF	ON
CLKOK	Enable	Enable CLKOK output clock	ON/OFF	OFF
	Bypass	Bypass PLL: CLKOK = CLKI	ON/OFF	OFF
	Desired Frequency	Without external capacitor	0.195 MHz to 210 MHz	50 MHz
		With external capacitor	0.016 MHz to 25 MHz[3]	—
	Divider	CLKOK Divider Setting (Divider Mode)	2 to 128 (all even numbers)	2
	Tolerance	CLKOK tolerance users can tolerate	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0	0.0
	Actual Frequency	Actual frequency achievable, Read only	—	—
PLL Phase & Duty Option		Dynamic/Static Mode selection	Dynamic Mode/Static Mode	Static Mode
Delay Adjust		Dynamic/Static/No delay selection	Dynamic/Static/No Delay	No Delay[4]
Provide PLL Reset		Provide PLL Reset Port	ON/OFF	OFF
Provide CLKOK Divider Reset		Provide CLKOK Reset Port	ON/OFF	OFF
Import LPC to ispLEVER project		Import .lpc file to ispLEVER project	ON/OFF	OFF

1. These values apply to SPLL. All other values apply to both GPLL and SPLL.
2. Phase Detector Input Frequency range 2 MHz to 50MHz.
3. For f_{IN} < 5MHz, f_{OUT_max} = 10 * f_{IN}.
4. IPexpress gives the user the ability to select the GPLL, SPLL, or to let the software choose, based upon the settings in the Delay Adjust section.

Frequency Calculation

Table 10-8 illustrates the Frequency limits at the Phase Detector inputs. Users must select CLKI Divider and CLKFB Divider values so that the Phase Detector Frequency falls within the range.

Let M = CLKI divider value
N = CLKFB divider value
V = CLKOP divider value

The basic equations are:

CLKOP Frequency = CLKI Frequency * N/M
f_{VCO} (VCO Frequency) = CLKOP Frequency * V
f_{PFD} (PFD Frequency) = CLKI Frequency / M = CLKFB Frequency (= CLKOP Frequency) / N

Example: If CLKI frequency is 25 MHz without external capacitor, the CLKI divider value can be only 1.

Table 10-8. Phase Detector Frequency (f_{PFD}) Range

PLL Type	External Capacitor	Frequency Range
GPLL	Without external capacitor	25 MHz to 420 MHz
	With external capacitor	2 MHz to 50 MHz[1]
SPLL	Without external capacitor	33 MHz to 420 MHz
	With external capacitor	2 MHz to 50 MHz[1]

1. For f_{IN} < 5MHz, f_{OUT_max} = 10 * f_{IN}.

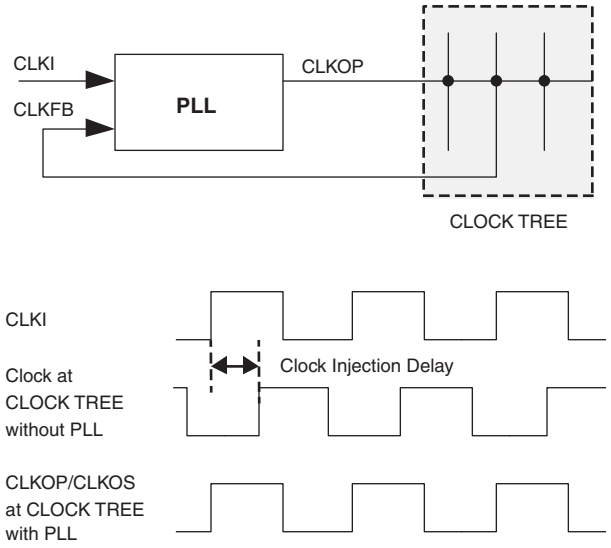
PLL Modes of Operation

PLLs have many uses within logic design. The two most popular are clock injection removal and clock phase adjustment. These two modes of operation are described below.

PLL Clock Injection Removal

In this mode, the PLLs are used to reduce clock injection delay. Clock injection delay is the delay from the input pin of the device to a destination element such as a flip-flop. The phase detector of the PLL aligns the CLKI with CLKFB. If the CLKFB signal comes from the clock tree (CLKOP), then the PLL delay and the clock tree delay is removed. Figure 10-16 Illustrates an example block diagram and waveform.

Figure 10-16. Clock Injection Delay Removal Application

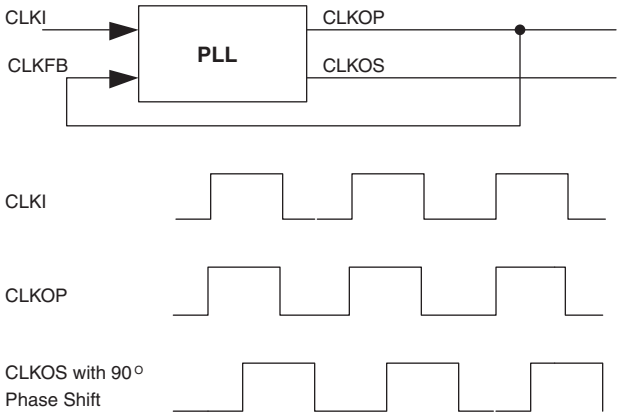


PLL Clock Phase Adjustment

In this mode, the PLLs are used to create fixed phase relationships in 22.5° increments. Creating fixed phase relationships are useful for forward clock interfaces where a specific relationship between the clock and data is required.

The fixed phase relationship can be used between CLKI and CLKOS or between CLKOP and CLKOS.

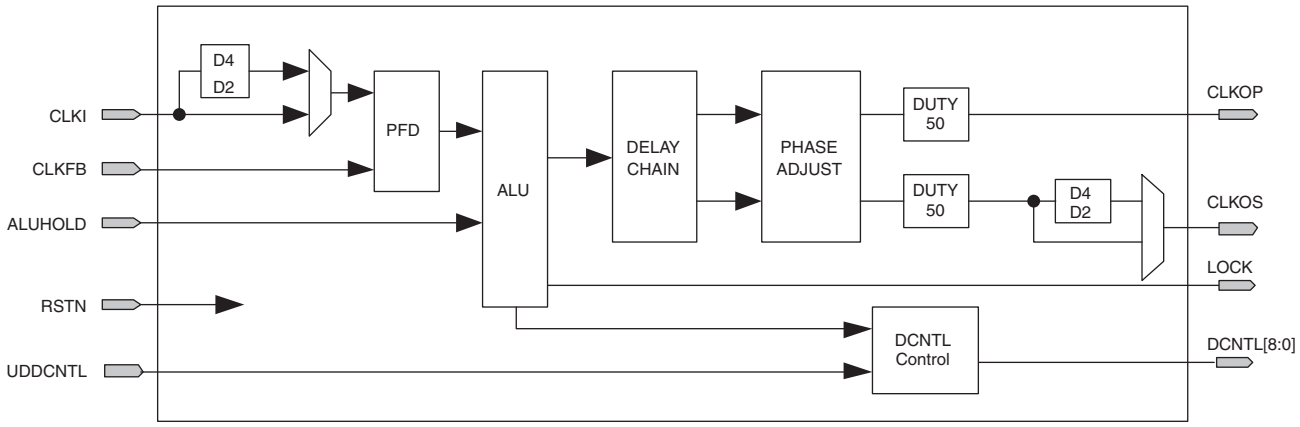
Figure 10-17. CLKOS Phase Adjustment from CLKOP



sysCLOCK DLL

The LatticeECP2/M DLL provides features such as clock injection delay removal, delay match, time reference delay (90° phase delay), and output phase adjustment. The DLL performs clock manipulation by adding delay to the CLKI input signal to create specific phase relationships. There are two types of outputs of the DLL. The first are clock signals similar to the PLL CLKOP and CLKOS. The other type of output is a delay control vector (DCNTL[8:0]). The delay control vector is connected to a Slave Delay Line (DLLDEL) element located in the I/O logic which matches the delay cells in the DLL. This delay vector allows the DLL to dynamically delay an input signal by a specific amount. Figure 10-18 provides a block diagram of the LatticeECP2/M DLL.

Figure 10-18. LatticeECP2/M DLL Block Diagram



Both clock injection delay removal and output phase adjustment use only the clock outputs of the DLL. Time reference delay and delay match modes use the delay control vector output. Specific examples of these features are discussed later in this document.

DLL Overview

The LatticeECP2/M DLL is created and configured by IPexpress. The following is a list of port names and descriptions for the DLL. There are two library elements used to implement the DLL: CIDDLLA (Clock Injection Delay), and TRDDLLA (Time Reference Delay). IPexpress will wrap one of these library elements to create a customized DLL module based on user selections.

DLL Inputs and Outputs

CLKI Input

The CLKI signal is the reference clock for the DLL. The CLKI input can be sourced from any type of FPGA routing and pin. The DLL CLKI input has a preferred pin per DLL which provides the lowest latency and best case performance.

CLKFB Input

The CLKFB input is available only if the user chooses to use a user clock signal for the feedback or in clock delay match mode. If internal feedback or CLKOS/CLKOP is used for the feedback, this connection will be made inside the module. In Clock Injection Delay Removal mode, the DLL will align the input clock phase with the feedback clock phase by delaying the input clock.

In Clock Injection Delay Match mode, the DLL will calculate the delta between the CLKI and CLKFB signals. This delay value is then output on the DCNTL vector. The DLL CLKFB input has a preferred pin per DLL which is discussed later in this document. The preferred pin provides the lowest latency and best case performance.

CLKOP Output

An output of the DLL based on the CLKI rate. The CLKOP output can drive primary and edge clock routing.

CLKOS Output

An output of the PLL based on the CLKI rate which can be divided and/or phase shifted. The CLKOS output can drive the primary and edge clock routing.

DCNTL[8:0] Output

This output of the DLL is used to delay a signal by a specific amount. The DCNTL[8:0] vector connects to a Slave Delay Line element. The DLL can then control multiple input delays from a single DLL.

UDDCNTL Input
This input is used to enable or disable updating of the DCNTL[8:0]. To ensure that the signal is captured by the synchronizer in the DLL block, it must be driven high for a time equal to at least two clock cycles when an update is required. If the signal is driven high and held in that state, the DCNTL[8:0] outputs are continuously updated.

ALUHOLD Input
This active high input stops the DLL from adding and subtracting delays to the CLKI signal. The DCNTL[8:0], CLKOP, and CLKOS outputs will still be valid, but will not change from the current delay setting.

LOCK Output
Active high lock indicator output. The LOCK output will be high when the CLKI and CLKFB signal are in phase. If the CLKI input stops the LOCK output will remain asserted. The clock is stopped so there is no clock to de-assert the LOCK output. Note that this is different from the operation of the PLL where the VCO continues to run when the input clock stops.

RSTN
Active low reset input to reset the DLL. The DLL can optionally be reset by the GSRN as well. It is recommended that if the DLL requires a reset, the reset should not be the same as the FPGA logic reset. Typically, logic requires that a clock is running during a reset condition. If the data path reset also resets the DLL, the source of the logic clock will stop and this may cause problems in the logic.

DLL Attributes

The LatticeECP2/M DLL utilizes several attributes that allow the configuration of the DLL through source constraints, IPexpress and preference files. The following section details these attributes and their usage.

DLL Lock on Divide by 2 or Divide by 4 CLKOS Output
Usually, the DLL is a ‘times one’ device, allowing neither frequency multiplication or division. But the LatticeECP2/M DLL allows ‘divide by 2’ or ‘divide by 4’ CLKOS outputs. Two optional ‘divide by 2’ and ‘divide by 4’ blocks are placed at the CLKI input as well as the CLKOS and this enables the use of divided CLKOS in the DLL feedback path. This allows the DLL to perform clock injection removal on a ‘divide by 2’ or ‘divide by 4’ clock, which is useful for DDRX2 and DDRX4 modes of I/O buffer operation.

When this optional clock divider is used only in the CLKOS output path, it allows the DLL to output two time-aligned clocks at different frequencies. When the divider is set to divide by 2 or divide by 4, a ‘dummy’ delay is inserted in the CLKOP output path to match the clock to Q delay of the CLKOS divider.

Optional Clock Fine Phase Shift
The optional fine phase shift in the CLKOS output path is built from a delay block that matches the other four blocks in the main delay chain. This delay block allows the CLKOS output to phase shifted a further 45, 22.5 or 11.25 degrees relative to its normal position.

GSR
The DLL can be reset by the GSR, if enabled. The GSR keyword can be set to ENABLED/DISABLED. This option is provided in the IPexpress GUI. Below is an example of the use of this preference.

```
ASIC “dll/dll_0_0” TYPE “CIDDLLA” GSR=DISABLED;
```

DLL Lock Time Control
The DLL will lock when the CLKI and CLKFB phases are aligned. In a simulation environment, the lock time is fixed to 100µs (default). This value can be changed through an HDL parameter or preference (for the back annotation simulation). The DLL contains a parameter named LOCK_DELAY which accepts an integer value for the total time in µs until the lock output goes high. Below is an example of how to set this value for front-end simulation.

Verilog:

```
defparam mydll.mypll_0_0.LOCK_DELAY=500;
mydll dll_inst(.CLKI(clkin), .CLKOP(clk1), .CLKOS(clk2),
```

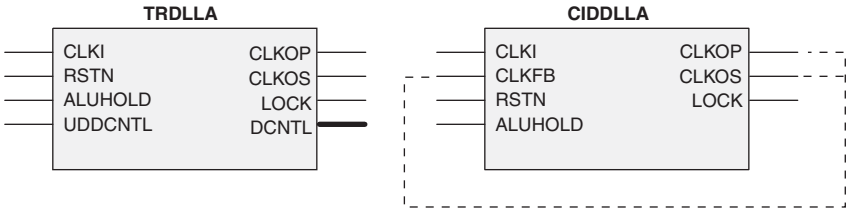
VHDL:

Not supported. For back annotation simulation LOCK_DELAY needs to be set in the preference file. Below is an example for the PLL.

```
ASIC "pll/pll_0_0" TYPE "EHXPLLA" LOCK_DELAY=200;
```

DLL Library Symbols

Figure 10-19. DLL Library Symbols



DLL Library Definitions

The Lattice library contains library elements to allow designers to utilize the DLL. These library elements use the DLL attributes defined in the "DLL Attributes" section.

The two modes of operation are presented as library elements as listed below.

Table 10-9. DLL Library Elements

Library Element Name	Mode of Operation	Description
TRDLLA	Time Reference Delay DLL	This mode generates four phases of the clock, 0°, 90°, 180°, 270°, along with the control setting used to generate these phases.
CIDDLLA	Clock Injection Delay DLL (Four Delay Cell Mode)	This mode removes the clock tree delay, aligning the external feedback clock to the reference clock. It has a single output coming from the fourth delay block.

DLL Library Element I/Os

Table 10-10. DLL Library Element I/O Descriptions

Signal	I/O	Description
CLKI	I	Clock input pin from dedicated clock input pin, other I/O or logic block.
CLKFB	I	Clock feedback input pin from dedicated feedback input pin, internal feedback, other I/O or logic block. This signal is not user selectable.
RSTN	I	Active low synchronous reset. From dedicated pin or internal node.
ALUHOLD	I	"1" freezes the ALU. For TRDLLA and CIDDLLA.
UDDCNTL	I	Active high synchronous enable signal from CIB for updating digital control to PIC delay. It must be driven high at least two clock cycles.
DCNTL[8:0]	O	Digital delay control code signals.
CLKOP	O	The primary clock output for all possible modes.
CLKOS	O	The secondary clock output with finer phase shift and/or division by 2 or by 4.
LOCK	O	Active high phase lock indicator. Lock means the reference and feedback clocks are in phase.

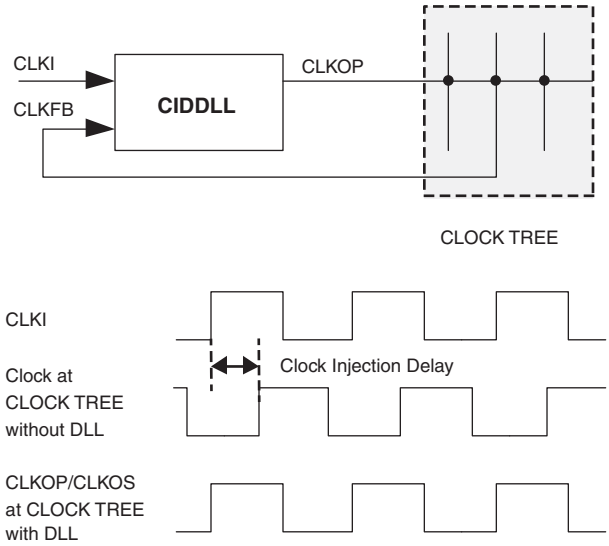
Note: Refer to device data sheet for frequency specifications.

DLL Modes of Operation

Clock Injection Removal Mode (CIDDLLA)

The DLL can be used to reduce clock injection delay (CIDDLLA). Clock injection delay is the delay from the input pin of the device to a destination element such as a flip-flop. The DLL will add delay to the CLKI input to align CLKI to CLKFB. If the CLKFB signal comes from the clock tree (CLKOP, CLKOS) then the delay of the DLL and the clock tree will be removed from the overall clock path. Figure 10-20 shows a circuit example and waveform.

Figure 10-20. Clock Injection Delay Removal via DLL



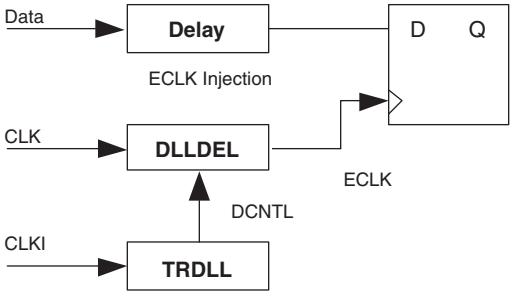
Clock injection removal mode can also provide a DCNTL port. In this mode, the delay added to the CLKI signal is output on the DCNTL port so that other input signals can be delayed by the same amount. This is very useful if several clocks are used in the same circuit to minimize the number of DLLs required. When using the DCNTL, the DLL delay will be limited to the range of the DCNTL vector. Therefore, IPexpress will restrict the CLKI rate from 300MHz to 700MHz.

Time Reference Delay Mode (TRDLLA: 90-Degree Phase Delay)

The Time Reference Delay (TRDDLLA) mode of the DLL is used to calculate 90 degrees of delay to be placed on the DCNTL vector. This is a useful mode in delaying a clock 90 degrees for use in clocking a DDR type interface.

Figure 10-21 provides a circuit example of this mode.

Figure 10-21. Time Reference Delay Circuit Example



In this mode, CLKI accepts a clock input. The DLL produces a DCNTL vector that will delay an input signal by 90 degrees of a full period of the CLKI signal. This DCNTL vector can then be connected to a Slave Delay Line (DLLDELA) to delay the signal by 90 degrees of the full period of CLKI.

DLL Usage in IPexpress

IPexpress is used to create and configure a DLL. The IPexpress graphical user interface allows users to select parameters for the DLL. The result is an HDL model to be used in the simulation and synthesis flow.

Configuration Tab

- **Usage Mode** – Select the mode of the DLL (Time Reference Delay [TRDLLA] or Clock Injection Delay Removal [CIDDLLA]). This selection will enable or disable further options in the GUI.
- **CLKI Frequency** – The rate of the CLKI input in MHz.
- **CLKOS Divider** – Set the divider for the CLKOS output to be either no divide, divide by 2, or divide by 4.
- **CLKOS Phase Shift** – Set the phase offset of the CLKOS to the CLKOP output. CLKOP will lead CLKOS by the amount of phase shift selected. The phase increment is 11.25 degrees. The pull-down list numbers are abbreviated to a decimal point.
- **CLKFB Feedback Mode** – Sets the feedback mode of the DLL to either CLKOP, CLKOS or User Clock. If CLKOP/CLKOS is selected, the clock tree injection delay for the specific output clock will be removed. If User Clock is selected, the user will be provided with the CLKFB port on the DLL.
- **CLKFB Frequency** – This is used only with User Clock Feedback Mode.
- **Provide RSTN Port** – The RSTN port allows the user to reset the DLL through a user signal.

PLL/DLL Cascading

It is possible to connect several arrangements of PLLs and DLLs. There are three possible cascading schemes:

- PLL to PLL
- PLL to DLL
- DLL to DLL

It is not possible to connect the DLL to a PLL. The DLL produces abrupt changes on its output clocks when changing delay settings. The PLL sees this as radical phase changes that prevent the PLL from locking correctly.

IPexpress Output

There are two outputs of IPexpress that are important for use in the design. The first is the <module_name>.[v|vhd] file. This is the user-named module that was generated by the tool to be used in both synthesis and simulation flows. The second file is a template file <module_name>_tmpl.[v|vhd]. This file contains a sample instantiation of the module. This file is only provided for the user to copy/paste the instance and is not intended to be used in the synthesis or simulation flows directly.

For the PLL/DLL, IPexpress sets attributes in the HDL module created that are specific to the data rate selected. Although these attributes can easily be changed, they should only be modified by re-running the GUI so that the performance of the PLL/DLL is maintained. After the map stage in the design flow, FREQUENCY preferences will be included in the preference file to automatically constrain the clocks produced from the PLL/DLL.

Clock Dividers (CLKDIV)

The clock divider divides the high-speed clock by 2, 4 and 8. The divided output can then be used as a primary clock or secondary clock input. The clock dividers are used for providing the low-speed FPGA clocks for shift registers (x2, x4, x8) and DDR/SPI4 I/O logic interfaces. There are two clock dividers, one on each side.

CLKDIV Library Element Definition

Users can instantiate CLKDIV in the source code as defined in this section. Figure 10-22, Table 10-11 and Table 10-12 describe the definition of CLKDIVB.

Figure 10-22. CLKDIV Library Symbol

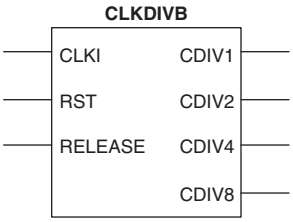


Table 10-11. CLKDIVB Port Definitions

Name	Description
CLKI	Clock input
RST[1]	Reset input, asynchronously forces all outputs low.
RELEASE[1]	Releases outputs synchronously to input clock.
CDIV1	Divided by 1 output
CDIV2	Divided by 2 output
CDIV4	Divided by 4 output
CDIV8	Divided by 8 output

1. Note: Unused RST must be tied to ground. Unused RELEASE must be tied to V_{CC}.

Table 10-12. CLKDIVB Attribute Definition

Name	Description	Value	Default
GSR	GSR Enable	ENABLED/DISABLED	DISABLED

CLKDIV Declaration in VHDL Source Code

```
COMPONENT CLKDIVB
-- synthesis translate_off
         GENERIC (
         GSR             : in String);
-- synthesis translate_on
         PORT (
         CLKI,RST, RELEASE:IN    std_logic;
         CDIV1, CDIV2, CDIV4, CDIV8:OUT   std_logic);
END COMPONENT;

     attribute GSR : string;
     attribute GSR of CLKDIVinst0 : label is “DISABLED”;

begin

CLKDIVinst0:CLKDIVB
-- synthesis translate_off
         GENERIC MAP(
         GSR             => “disabled”
         );
-- synthesis translate_on
         PORT MAP(
         CLKI            => CLKIsig,
         RST             => RSTsig,
         RELEASE         => RELEASEsig,
```

```
        CDIV1           => CDIV1sig,
        CDIV2           => CDIV2sig,
        CDIV4           => CDIV4sig,
        CDIV8           => CDIV8sig
        );

        end
```

CLKDIV Usage with Verilog - Example

```
module clkdiv_top(RST,CLKI,RELEASE,CDIV1,CDIV2,CDIV4,CDIV8);

input CLKI,RST,RELEASE;
output CDIV1,CDIV2,CDIV4,CDIV8;

CLKDIVB CLKDIBinst0 (.RST(RST),.CLKI(CLKI),.RELEASE(RELEASE),
             .CDIV1(CDIV1),.CDIV2(CDIV2),.CDIV4(CDIV4),.CDIV8(CDIV8));

defparam CLKDIBint0.GSR = "DISABLED";

endmodule
```

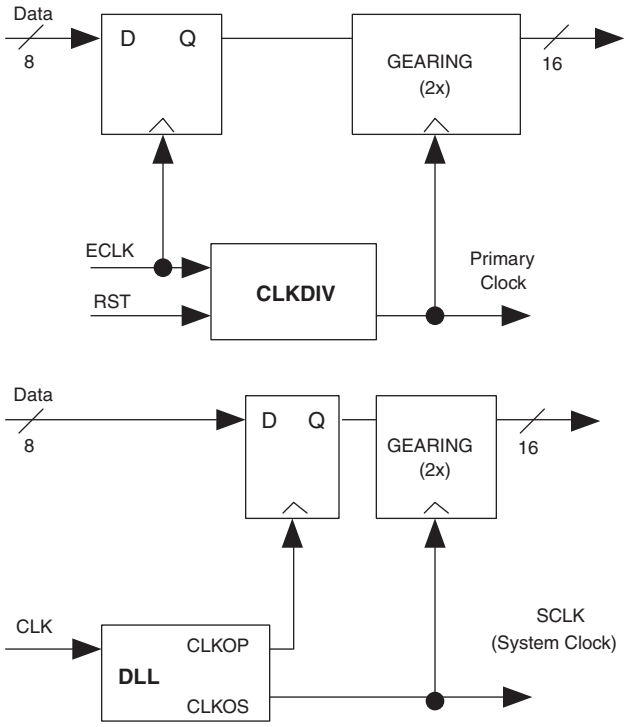
CLKDIV Example Circuits

The clock divider (CLKDIV) can divide a clock by 2 or 4 and drives a primary clock network. The clock dividers are useful for providing the low-speed FPGA clocks for I/O shift registers (x2, x4) and DDR (x2, x4) I/O logic interfaces. Divide by 8 is provided for slow speed/low power operation.

To guarantee a synchronous transfer in the I/O logic, the CLKDIV input clock must come from an edge clock and the output drives a primary clock. In this case, they are phase matched. This is especially useful for synchronously resetting the I/O logic when Mux/DeMux gearing is used in order to synchronize the entire data bus as shown in Figure 10-23. Using the low skew characteristics of the edge clock routing, a reset can be provided to all bits of the data bus to synchronize the Mux/DeMux gearing.

The second circuit shows that a DLL can replace CLKDIV for x2 and x4 applications.

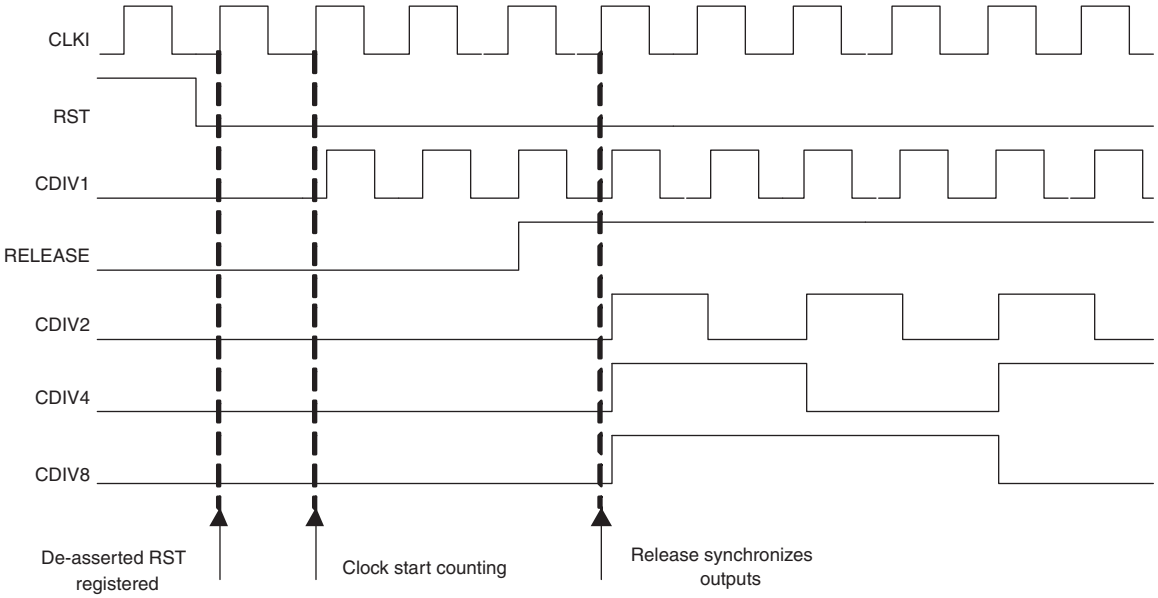
Figure 10-23. CLKDIV Application Examples



Release Behavior

The port “Release” is used to synchronize all outputs after RST is de-asserted. Figure 10-24 illustrates the Release behavior.

Figure 10-24. CLKDIV Release Behavior



DLLDEL (Slave Delay Line)

The Slave Delay line is designed to generate the desired delay in DDR/SPI4 applications. The delay control inputs (DCNTL[8:0]) are fed from the general purpose DLL outputs. The library element definitions are described in Figure 10-25 and Table 10-13.

Figure 10-25. DLLDELA Library Symbol

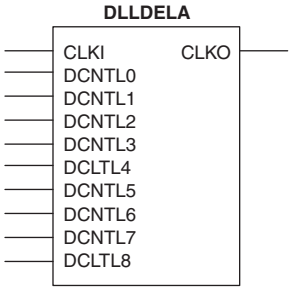


Table 10-13. DLLDELA I/O

Name	I/O	Description
CLKI	I	Clock Input
DCNTL[8:0]	I	Delay Control Bits
CLKO	O	Clock Output

DLLDELA Declaration in VHDL Source Code

```
COMPONENT DLLDELA
      PORT     (
               CLKI :IN std_logic;
               DCNTL0 :IN std_logic;
               DCNTL1 :IN std_logic;
               DCNTL2 :IN std_logic;
               DCNTL3 :IN std_logic;
               DCNTL4 :IN std_logic;
               DCNTL5 :IN std_logic;
               DCNTL6 :IN std_logic;
               DCNTL7 :IN std_logic;
               DCNTL8 :IN std_logic;
               CLKO :OUT std_logic
               );
END COMPONENT;

begin
DLLDELAinst0: DLLDELA1
      PORT MAP (
               CLKI => clkisig,
               DCNTL0 => dcntl0sig,
               DCNTL1 => dcntl1sig,
               DCNTL2 => dcntl2sig,
               DCNTL3 => dcntl3sig,
               DCNTL4 => dcntl4sig,
               DCNTL5 => dcntl5sig,
               DCNTL6 => dcntl6sig,
               DCNTL7 => dcntl7sig,
               DCNTL8 => dcntl8sig,
```

```
                CLKO => clkosig
                );

   end
```

DLLDELA Usage with TRDLLA - Verilog - Example

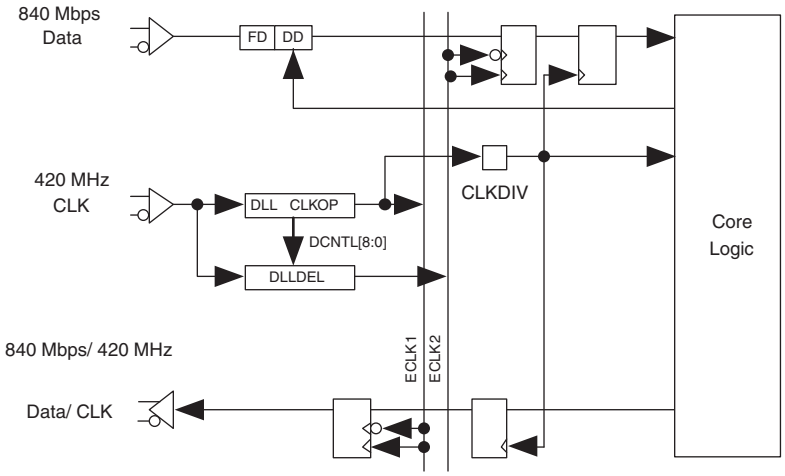
Note: DLL0(TRDLLA) must be generated by IPexpress as a sub-module

```
module ddldel_top (rst,d,clkin,clkin2,clkout,aluhold,uddcntl,q);

input rst,d,clkin,clkin2,aluhold,uddcntl;
output clkout,q;

wire [8:0]DCntl_int;
reg qint;

DLL0   dllinst0   (.clk(clkin2),   .aluhold(aluhold),   .uddcntl(uddcntl),   .clkop(),
.clkos(),
          .dcntl(DCntl_int),.lock());
DLLDELA delinst0 (.CLKI(clkin),.DCNTL0(DCntl_int[0]),.DCNTL1(DCntl_int[1]),
            .DCNTL2(DCntl_int[2]), .DCNTL3(DCntl_int[3]), .DCNTL4(DCntl_int[4]),
            .DCNTL5(DCntl_int[5]), .DCNTL6(DCntl_int[6]), .DCNTL7(DCntl_int[7]),
            .DCNTL8(DCntl_int[8]), .CLKO(clk90)); //synthesis syn_black_box

assign clkout = clk90;
assign q = qint;

always@(posedge clk90 or negedge rst)
      if (~rst)
        qint =1'b0;
      else
        qint = d;

endmodule
```

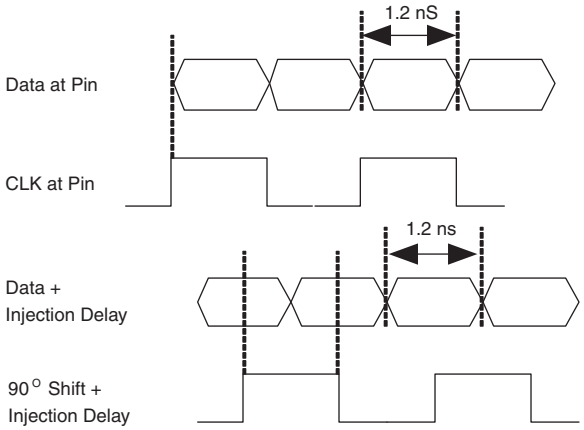
DLLDELA Application Example

Figure 10-26 shows an example DLLDEL application. As shown in the timing diagram, DLLDEL shifts the clock by 90 degrees to center both edges in the middle of data window.

Figure 10-26. SPI4.2 and DDR Registers Interface Application



FD: Fixed Delay
DD: Dynamic Delay
Users can select the delay setting in IPexpress.



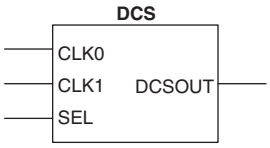
DQSDLL and DQSDEL

There is another combination of DLL and Slave Delay Line, DQSDLL and DQSDEL, in the LatticeECP2/M device family. This pair is similar in design and function to DLL and DLLDEL, but usage is limited to DDR implementation. For additional information, see TN1102, LatticeECP2/M sysIO Usage Guide.

DCS (Dynamic Clock Select)

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 where the enable signal is toggled. There are two DCSs for each quadrant. The outputs of the DCS then reach primary clock distribution via the feed-lines. Figure 10-27 shows the block diagram of the DCS.

Figure 10-27. DCS Library Symbol



DCS Library Element Definition

Table 10-14 defines the I/O ports of the DCS block. There are eight modes available. Table 10-15 describes how each mode is configured.

Table 10-14. DCS I/O Definition

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

Table 10-15. DCS Modes of Operations

Attribute Name	Description	Output		Value
		SEL = 0	SEL = 1	
DCS MODE	Rising edge triggered, latched state is high	CLK0	CLK1	POS
	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

DCS Timing Diagrams

Each mode performs its unique operation. Clock output timing is determined by input clocks and the edge of the SEL signal. Figure 10-28 describes the timing of each mode.

Figure 10-28. Timing Diagrams by DCS MODE

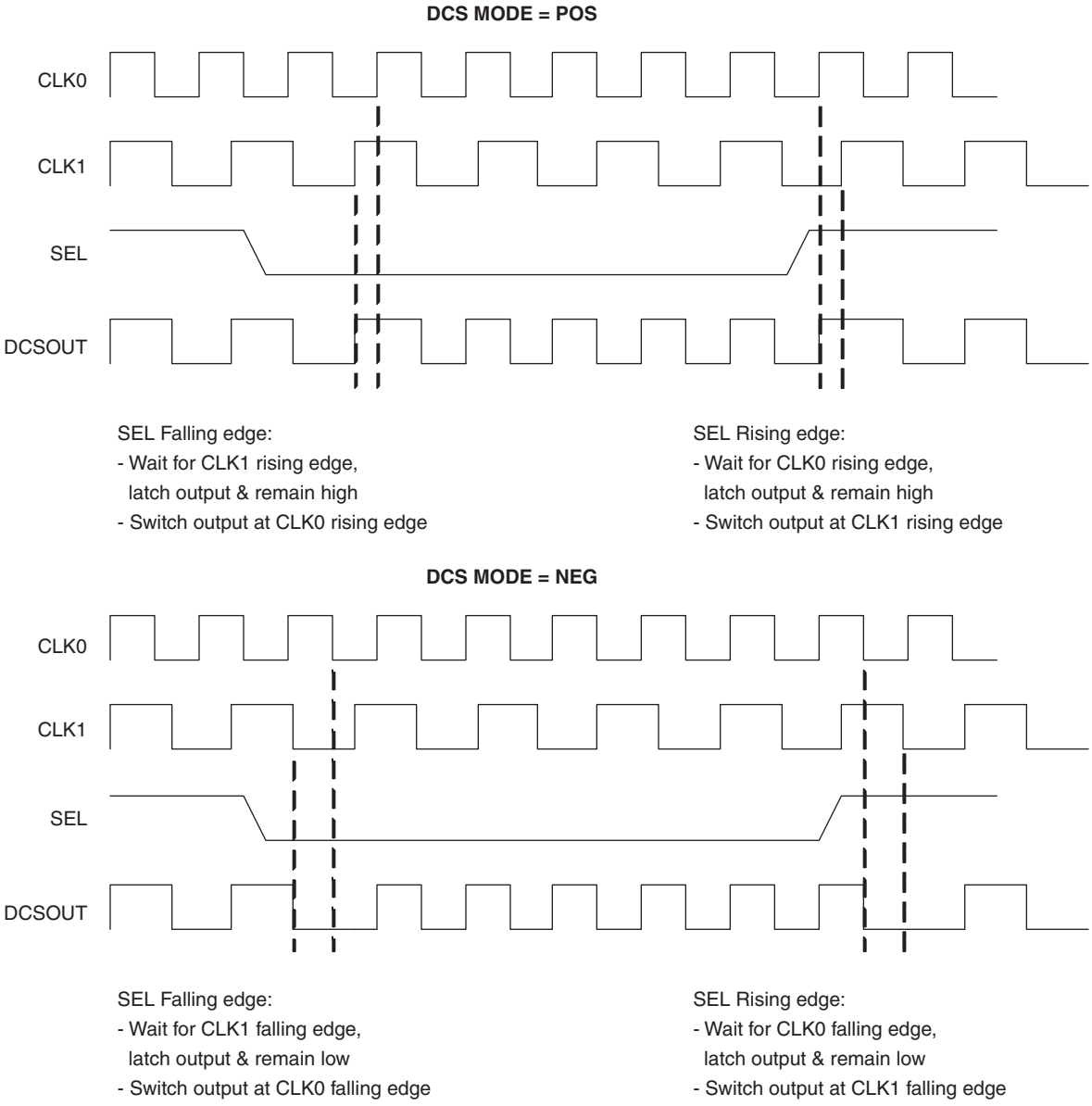
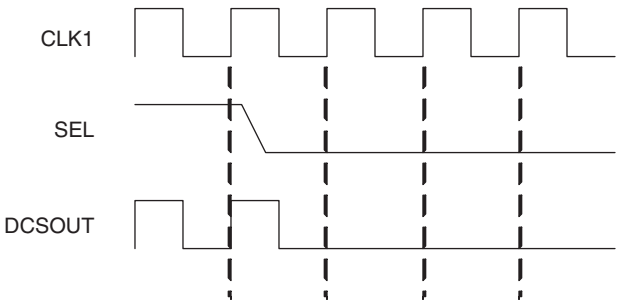


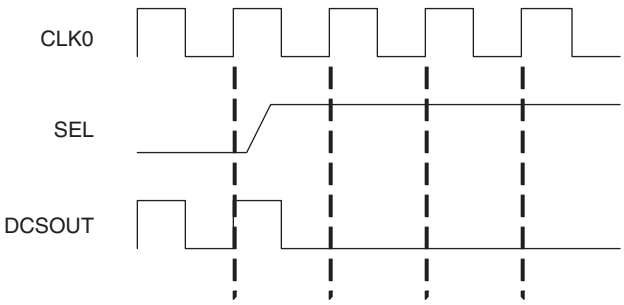
Figure 10-28. Timing Diagrams by DCS MODE (Cont.)

DCS MODE = HIGH_LOW



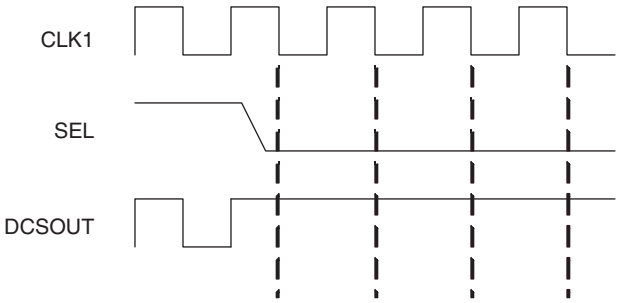
- Switch low @CLK1 falling edge.
- If SEL is low, output stays low at on CLK1 rising edge. SEL must not change during setup prior to rising clock.

DCS MODE = LOW_LOW



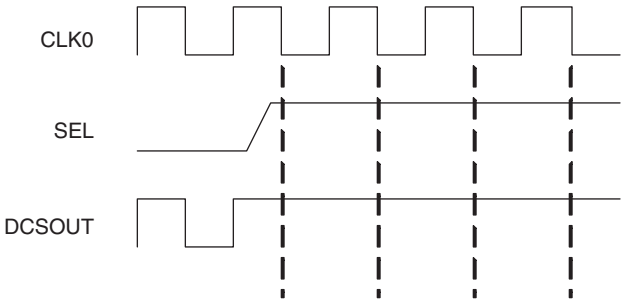
- Switch low @CLK0 falling edge.
- If SEL is high, output stays low at on CLK0 rising edge.

DCS MODE = HIGH_HIGH



- Switch high @CLK1 rising edge.
- If SEL is low, output stays low high on CLK1 falling edge.

DCS MODE = LOW_HIGH



- Switch high @ CLK0 rising edge.
- If SEL is high, output stays high on CLK0 falling edge.

DCS Usage with VHDL - Example

```
library ecp2m;
use ecp2m.components.all;
library IEEE;
use IEEE.std_logic_1164.all;

entity DCStest is port
( clksel  : in  std_logic;
  dcsclk0 : in  std_logic;
  sysclk1 : in  std_logic;
  dcsclk  : out std_logic
);
end DCStest;
architecture DCStest_arch of DCStest is
  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   => dcsclk0,
    CLK1   => sysclk1,
    DCSOUT => dcsclk
  );
end DCStest_arch;
```

DCS Usage with Verilog - Example

```
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
```

Oscillator (OSCD)

There is a dedicated oscillator in the LatticeECP2/M devices whose output is made available for users. The oscillator frequency is programmable with a range of 2.5 to 130MHz. The output of the oscillator can also be routed as an input clock to the clock tree. The oscillator frequency output can be further divided by internal logic (user logic) for lower frequencies, if desired. The oscillator is powered down when not in use. The output of this oscillator is not a precision clock. It is intended for use as an extra clock that does not require accurate clocking.

Library Element Name: OSCD

Table 10-16. OSCD Port Definition

I/O	Name	Description
Output	OSC	Oscillator Clock Output

Table 10-17. OSCD Attribute Definition

User Attribute	Attribute Name	Value (MHz)	Default Value
Nominal Frequency	NOM_FREQ	2.5, 4.3, 5.4, 6.9, 8.1, 9.2, 10.0, 13.0, 15.0, 20.0, 26.0, 30.0, 34.0, 41.0, 45.0, 55.0, 60.0, 130.0	2.5

Please refer to the LatticeECP2/M Family Data Sheet for detailed specifications.

OSC Library Symbol (OSCD)

Figure 10-29. OCS Symbol

OSC Usage with VHDL - Example

```
COMPONENT OSCD
-- synthesis translate_off
      GENERIC(NOM_FREQ: string);
-- synthesis translate_on
PORT (CFGCLK:OUT   std_logic);

END COMPONENT;

      attribute NOM_FREQ : string;
      attribute NOM_FREQ of OSCins0 : label is “2.5”;
begin

OSCInst0: OSCD
-- synthesis translate_off
      GENERIC MAP (NOM_FREQ => “2.5”)
-- synthesis translate_on
      PORT MAP ( CFGCLK=>  osc_int);

end
```

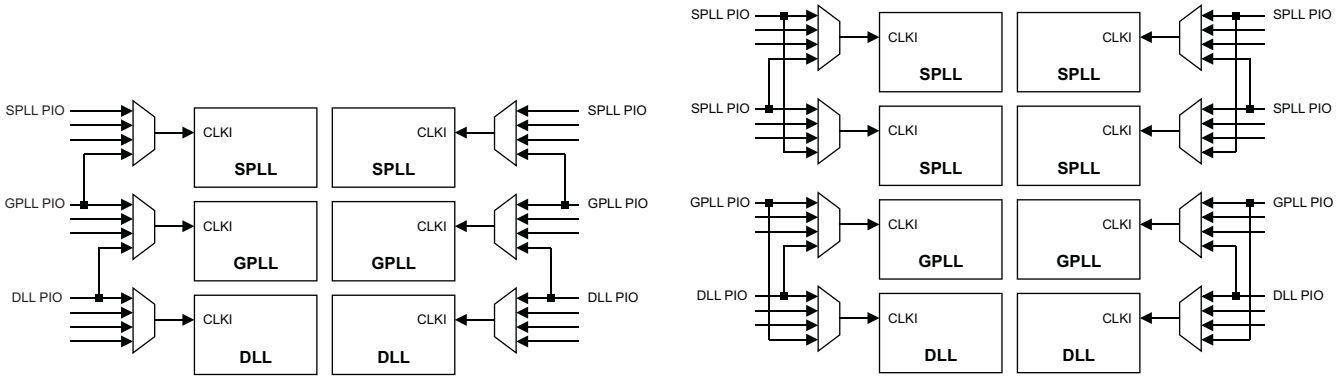
OSC Usage with Verilog - Example

```
module OSC_TOP(OSC_CLK);

output OSC_CLK;

OSCD OSCinst0 (.CFGCLK(OSC_CLK));

defparam OSCinst0.NOM_FREQ = "2.5";

endmodule
```

Input Clock Sharing

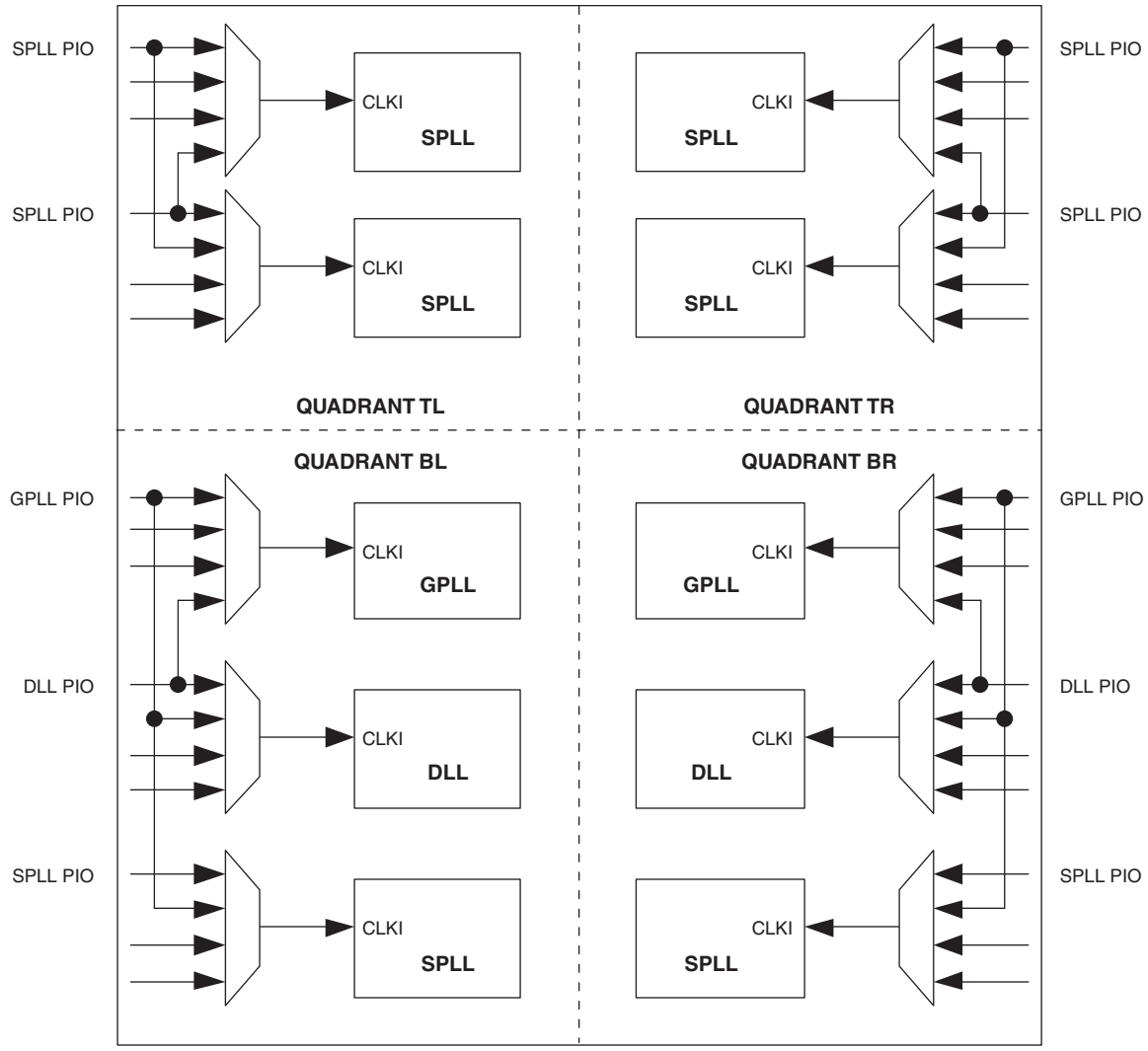
The reference clock from the pads can be shared in LatticeECP2/M PLLS and DLLS as shown in Figures 10-30 and 10-31. This feature is useful when only one clock source is available for multiple PLLs/DLLs.

Figure 10-30. Input Clock Sharing (LatticeECP2-50 and LatticeECP2-70)



LatticeECP2-50 **LatticeECP2-70**

Figure 10-31. Input Clock Sharing (LatticeECP2M with Six SPLLs)



Setting Clock Preferences

Clock preferences allow designers to implement clocks to the desired performance. Preferences can be set in the ispLEVER Design Planner Spreadsheet View (or **Tools > Spreadsheet View** in Diamond) or in preference files. Frequently used preferences are described in Appendix C. For additional information see the ispLEVER or Diamond on-line Help system.

Power Supplies

Each PLL has its own power supply. On the LatticeECP2-6, LatticeECP2-12, and LatticeECP2-20 devices the PLL power supply has been combined with the package VCC for better performance. There will only be VCC pins on these devices.

On the larger LatticeECP2 and all LatticeECP2M devices, the PLL power supply has its own power supply pins, VCCPLL. Since VCC and VCCPLL are normally the same 1.2V, it is recommended that they are driven from the same power supply on the circuit board, thus minimizing leakage. In addition, each of these supplies should be independently isolated from the main 1.2V supply on the board using proper board filtering techniques to minimize the noise coupling between them.

The DLL is powered from the FPGA core power supply.

Technical Support Assistance

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
February 2006	01.0	Initial release.
April 2006	01.1	Removed unsupported devices, removed DLL SMI phrases, rephrased DDUTY support due to software incomplete.
September 2006	01.2	Added detailed clock network descriptions.
		Added IPexpress GUI quick reference table.
		Added LatticeECP2M information throughout.
		OSC divider value range updated.
January 2007	01.3	Updated Frequency range.
		Described CLKOP and CLKOK synchronous timing relationship with respective reset signals.
January 2007	01.4	Updated IPexpress Main Window screen shot.
		Updated LatticeECP2/M Configuration Tab screen shot.
		Updated User Parameters in the Configuration GUI table.
June 2007	01.5	Corrected reference to EXHPLLD in Optional External Capacitor section (changed to EHXPLLD).
		Updated GSR section of Attributes.
August 2008	01.6	Updated the LatticeECP2/M PLL Configuration tab screen shot.
		Updated the Port names for the LatticeECP2/M PLL Library symbols.
		Added LatticeECP2/M PLL Modules section.
		Corrected the External Capacitor section.
		Removed the DLL operation mode CIMDLLA since it is not available on LatticeECP2/M.
		Updated Power Supplies section.
March 2009	01.7	Updated “DCS Usage with VHDL - Example” code.
October 2009	01.8	Updated Input Clock Sharing (LatticeECP2-50 and LatticeECP2-70) figure.
February 2010	01.9	Reconciled LOCK description among MachXO, LatticeXP2, LatticeECP2/M and LatticeECP3.
May 2010	02.0	Specified dedicated clock pins in the Secondary Clocks text section.
June 2010	02.1	Updated for Lattice Diamond design software support.
June 2013	02.2	Updated document with new corporate logo.
		Updated Technical Support Assistance information.

Appendix A. Primary Clock Sources and Distribution

Figure 10-32. LatticeECP2 Primary Clock Sources and Distribution

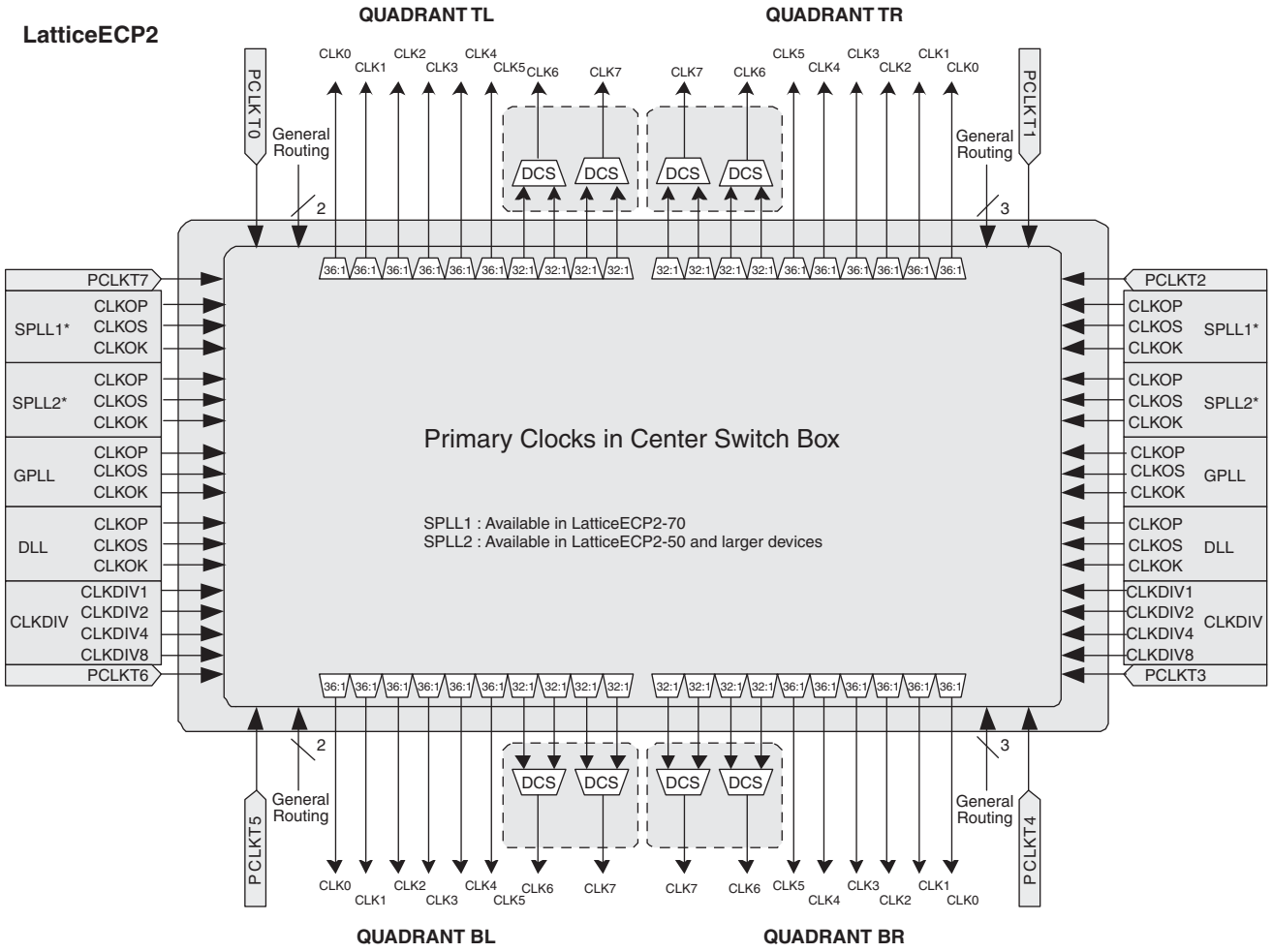


Figure 10-33. LatticeECP2 Primary Clock Muxes

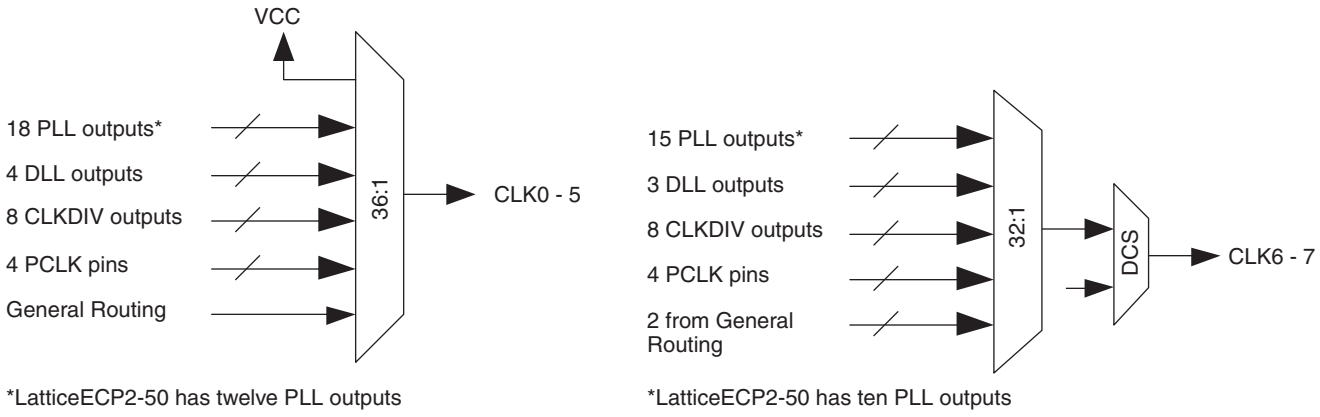


Figure 10-34. LatticeECP2M Primary Clock Sources and Distribution

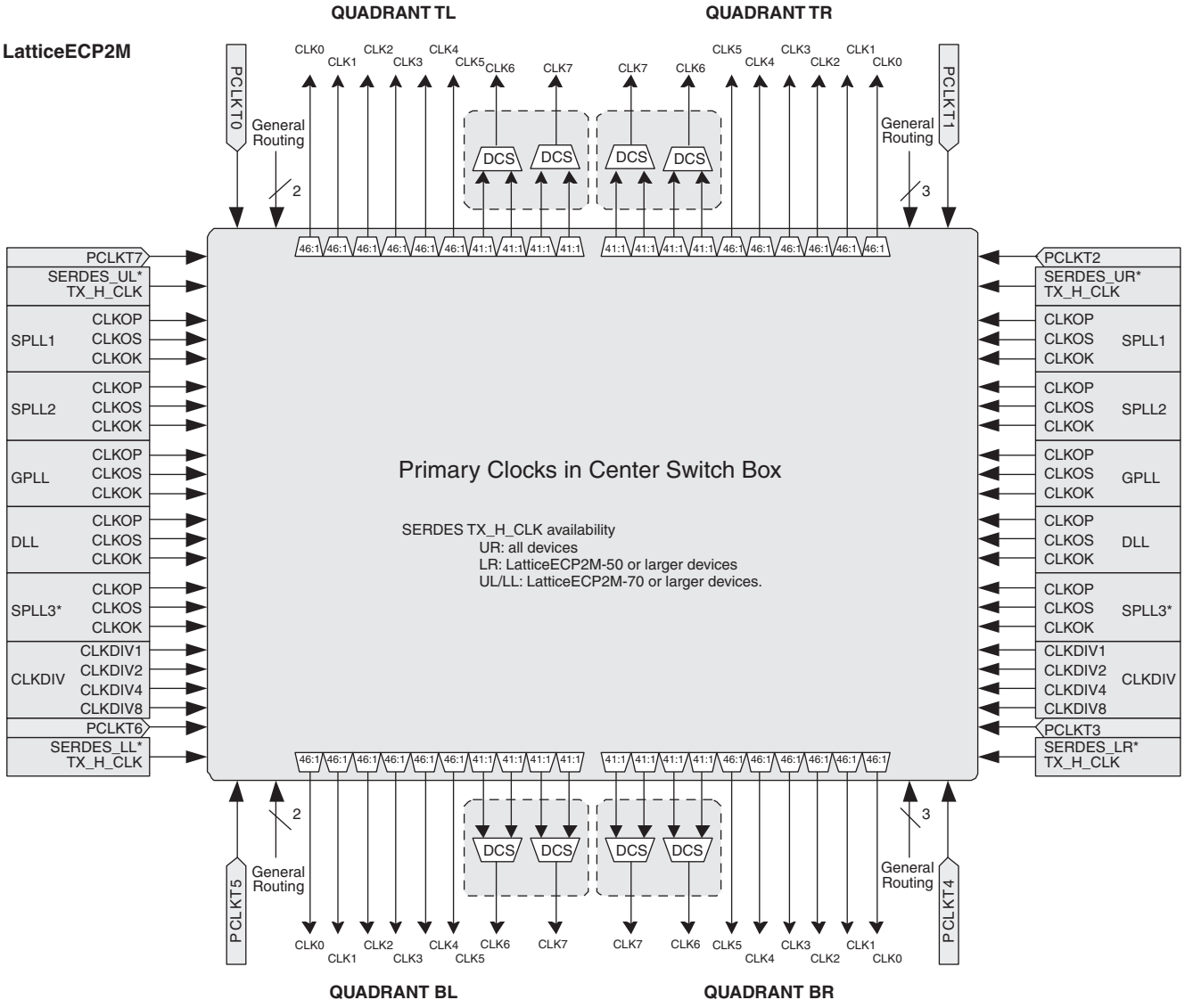
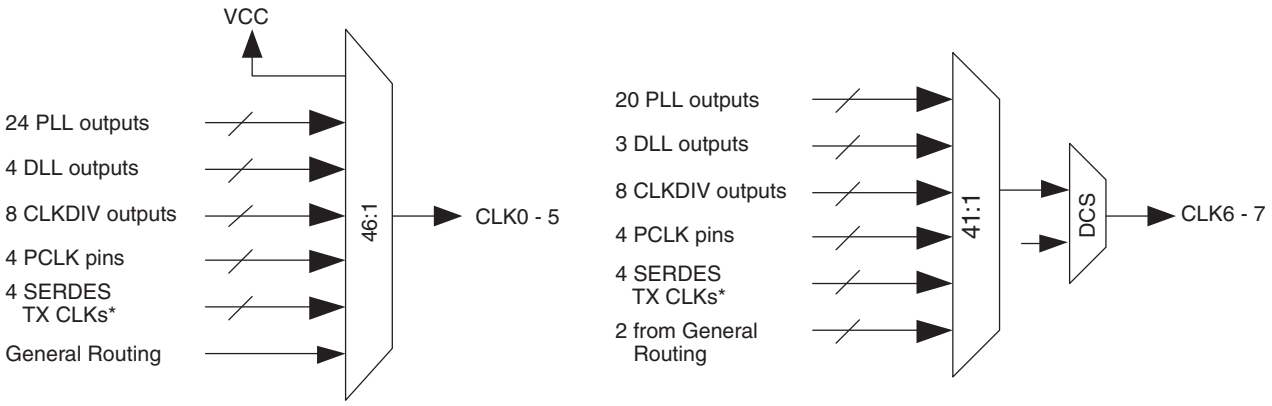


Figure 10-35. LatticeECP2M Primary Clock Muxes

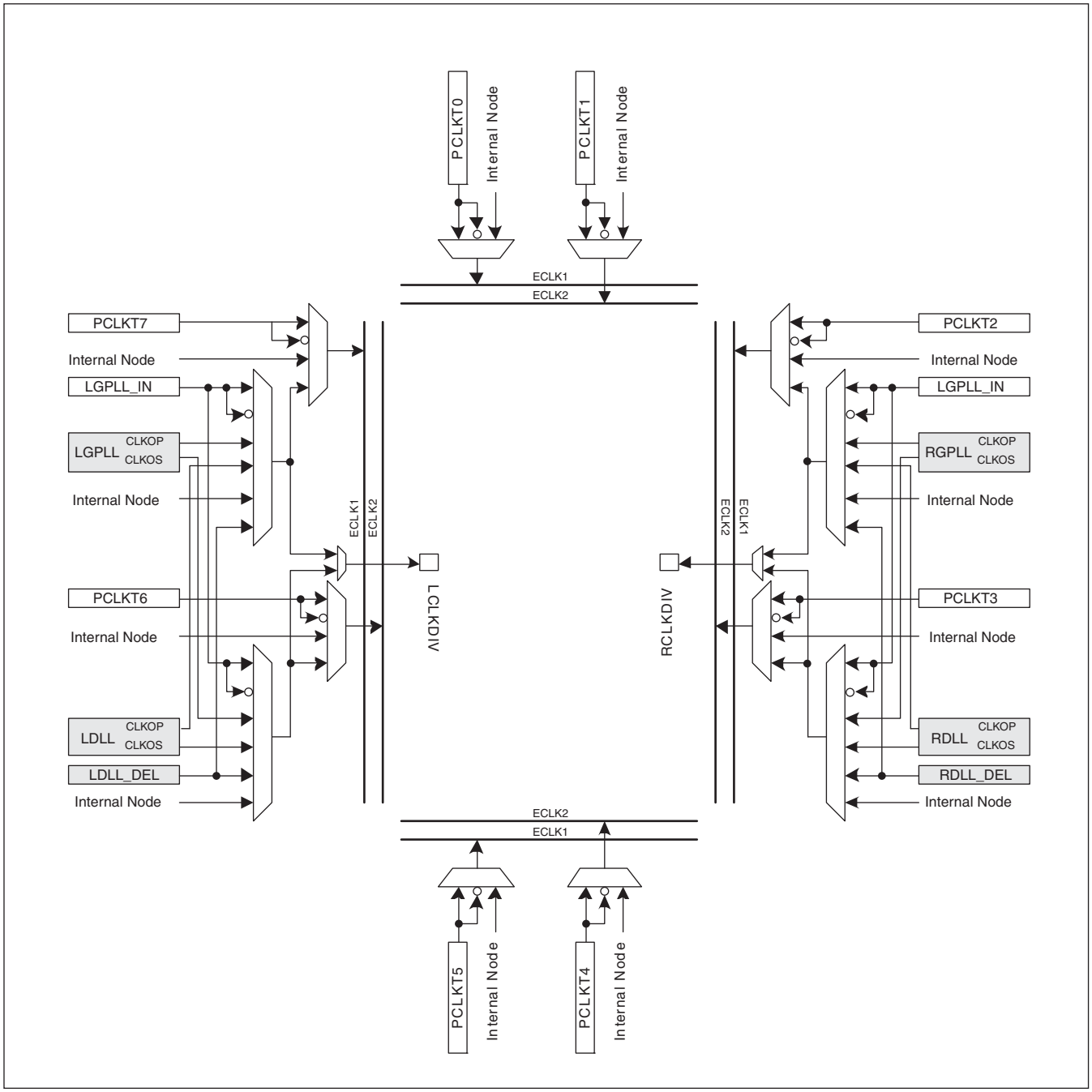


*LatticeECP2M-50 has two SERDES TX CLK outputs

Appendix B. PLL, DLL, CLKIDV and ECLK Locations and Connectivity

Figure 10-36 shows the locations, site names, and connectivity of the PLLs, DLLs, CLKDIVs and ECLKs

Figure 10-36. PLL, DLL, CLKIDV and ECLK Locations and Connectivity



Appendix C. Clock Preferences

A few key clock preferences are introduced below. Refer to the software “Help” file for other preferences and detailed information.

ASIC
The following preference command assigns a phase of 90° to the TRDLLA_CLKOS:

```
ASIC "my_dll" TYPE "TRDLLA" CLKOS_PHASE=90;
```

FREQUENCY
The following physical preference command assigns a frequency of 100 MHz to a net named clk1:

```
FREQUENCY NET "clk1" 100 MHz;
```

The following preference specifies a hold margin value for each clock domain:

```
FREQUENCY NET "RX_CLKA_CMOS_c" 100.000 MHz HOLD_MARGIN 1 ns;
```

MAXSKEW
The following command assigns a maximum skew of 5ns to a net named NetB:

```
MAXSKEW NET "NetB" 5 NS;
```

MULTICYCLE
The following command will relax the period to 50ns for the path starting at COMPA to COMPB (NET1):

```
MULTICYCLE "PATH1" START COMP "COMPA" END COMP "COMPB" NET "NET1" 50 NS ;
```

PERIOD
The following command assigns a clock period of 30ns to the port named Clk1:

```
PERIOD PORT "Clk1" 30 NS;
```

PROHIBIT
This command prohibits the use of a primary clock to route a clock net named bf_clk:

```
PROHIBIT PRIMARY NET "bf_clk";
```

USE PRIMARY
Use a primary clock resource to route the specified net.

```
USE PRIMARY NET clk_fast;
USE PRIMARY DCS NET "bf_clk";
USE PRIMARY PURE NET “bf_clk” QUADRANT_TL;
```

USE SECONDARY
Use a secondary clock resource to route the specified net.

```
USE SECONDARY NET "clk_lessfast" QUADRANT_TL;
```

USE EDGE
Use a edge clock resource to route the specified net.

```
USE EDGE NET “clk_fast”;
```

CLOCK_TO_OUT
Specifies a maximum allowable output delay relative to a clock.

Here are two preferences using both the CLKPORT and CLKNET keywords showing the corresponding scope of TRACE reporting.

The CLKNET will stop tracing the path before the PLL, so you will not get PLL compensation timing numbers.

```
CLOCK_TO_OUT PORT "RxAddr_0" 6.000000 ns CLKNET "pll_rxclk" ;
```

The above preference will yield the following clock path:

```
Physical Path Details:
     Clock path pll_inst/pll_utp_0_0 to PFU_33:
  Name     Fanout   Delay (ns)          Site               Resource
ROUTE         49     2.892    ULPPLL.MCLK to     R3C14.CLK0 pll_rxclk
                  --------
                    2.892   (0.0% logic, 100.0% route), 0 logic levels.
```

If CLKPORT is used, the trace is complete back to the clock port resource and provides PLL compensation timing numbers.

```
CLOCK_TO_OUT PORT "RxAddr_0" 6.000000 ns CLKPORT "RxClk" ;
```

The above preference will yield the following clock path:

```
Clock path RxClk to PFU_33:
  Name     Fanout   Delay (ns)          Site               Resource
IN_DEL       ---     1.431          D5.PAD to        D5.INCK RxClk
ROUTE          1     0.843        D5.INCK to   ULPPLL.CLKIN RxClk_c
MCLK_DEL     ---     3.605   ULPPLL.CLKIN to    ULPPLL.MCLK pll_inst/pll_utp_0_0
ROUTE         49     2.892    ULPPLL.MCLK to     R3C14.CLK0 pll_rxclk
                  --------
                    8.771   (57.4% logic, 42.6% route), 2 logic levels.
```

INPUT_SETUP
Specifies an setup time requirement for input ports relative to a clock net.

```
INPUT_SETUP PORT "datain" 2.000000 ns HOLD 1.000000 ns CLKPORT "clk"
PLL_PHASE_BACK ;
```

PLL_PHASE_BACK
This preference is used with INPUT_SETUP when the user needs a Trace calculation based on the previous clock edge.

This preference is useful when setting the PLL output Phase Adjustment. Since there is no negative phase adjustment provided, the PLL_PHASE_BACK preference works as if negative phase adjustment is available.

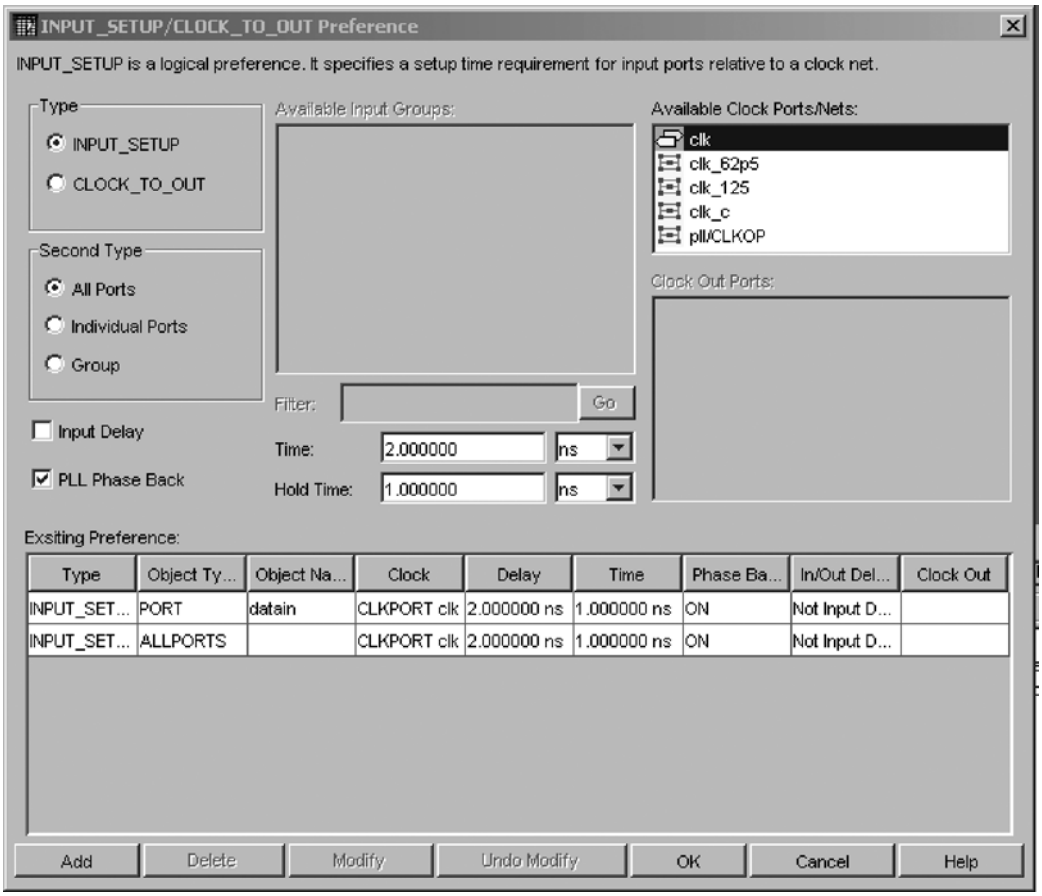
For example:

If a Phase Adjustment of -90° of CLKOS is desired, the user can set the Phase to 270° and set the INPUT_SETUP preference with PLL_PHASE_BACK.

PLL_PHASE_BACK Usage in Pre-Map Preference Editor: The Pre-Map Preference Editor can be used to set the PLL_PHASE_BACK attribute.

1. Open the Design Planner (Pre-Map).
2. In the Design Planner control window, select **Spreadsheet View** under **View**.
3. In the Spreadsheet View window, select **Input_setup/Clock_to_out…**
4. The INPUT_SETUP/CLOCK_TO_OUT Preference window is shown in Figure 10-37.

Figure 10-37. INPUT_SETUP/CLOCK_TO_OUT Preference Window (see Appendix D Figure 10-41 for Diamond Equivalent)



Appendix D. Lattice Diamond Usage Overview

This appendix discusses the use of Lattice Diamond design software for projects that include the LatticeECP2M SERDES/PCS module .

For general information about the use of Lattice Diamond, refer to the Lattice Diamond Tutorial.

If you have been using ispLEVER software for your FPGA design projects, Lattice Diamond may look like a big change. But if you look closer, you will find many similarities because Lattice Diamond is based on the same toolset and work flow as ispLEVER. The changes are intended to provide a simpler, more integrated, and more enhanced user interface.

Converting an ispLEVER Project to Lattice Diamond

Design projects created in ispLEVER can easily be imported into Lattice Diamond. The process is automatic except for the ispLEVER process properties, which are similar to the Diamond strategy settings, and PCS modules. After importing a project, you need to set up a strategy for it and regenerate any PCS modules.

Importing an ispLEVER Design Project

Make a backup copy of the ispLEVER project or make a new copy that will become the Diamond project.

1. In Diamond, choose **File > Open > Import ispLEVER Project**.
2. In the ispLEVER Project dialog box, browse to the project's .syn file and open it.
3. If desired, change the base file name or location for the Diamond project. If you change the location, the new Diamond files will go into the new location, but the original source files will not move or be copied. The Diamond project will reference the source files in the original location.

The project files are converted to Diamond format with the default strategy settings.

Adjusting PCS Modules

PCS modules created with IPexpress have an unusual file structure and need additional adjustment when importing a project from ispLEVER. There are two ways to do this adjustment. The preferred method is to regenerate the module in Diamond. However this may upgrade the module to a more recent version. An upgrade is usually desirable but if, for some reason, you do not want to upgrade the PCS module, you can manually adjust the module by copying its .txt file into the implementation folder. If you use this method, you must remember to copy the .txt file into any future implementation folders.

Regenerate PCS Modules

1. Find the PCS module in the Input Files folder of File List view. The module may be represented by an .lpc, .v, or .vhd file.
2. If the File List view shows the Verilog or VHDL file for the module, and you want to regenerate the module, import the module's .lpc file:
   a. In the File List view, right-click the implementation folder () and choose **Add > Existing File**.
   b. Browse for the module's .lpc file, ***<module_name>*.lpc**, and select it.
   c. Click **Add**. The .lpc file is added to the File List view.
   d. Right-click the module's Verilog or VHDL file and choose **Remove**.
3. In File List, double-click the module's .lpc file. The module's IPexpress dialog box opens.
4. In the bottom of the dialog box, click **Generate**. The Generate Log tab is displayed. Check for errors and close.

In File List, the .lpc file is replaced with an .ipx file. The IPexpress manifest (.ipx) file is new with Diamond. The .ipx file keeps track of the files needed for complex modules.

Using IPexpress with Lattice Diamond

Using IPexpress with Lattice Diamond is essentially same as with ispLEVER.

The configuration GUI tabs are all the same except for the Generation Options tab. Figure 10-38 shows the Generation Options tab window.

Figure 10-38. Generation Options Tab

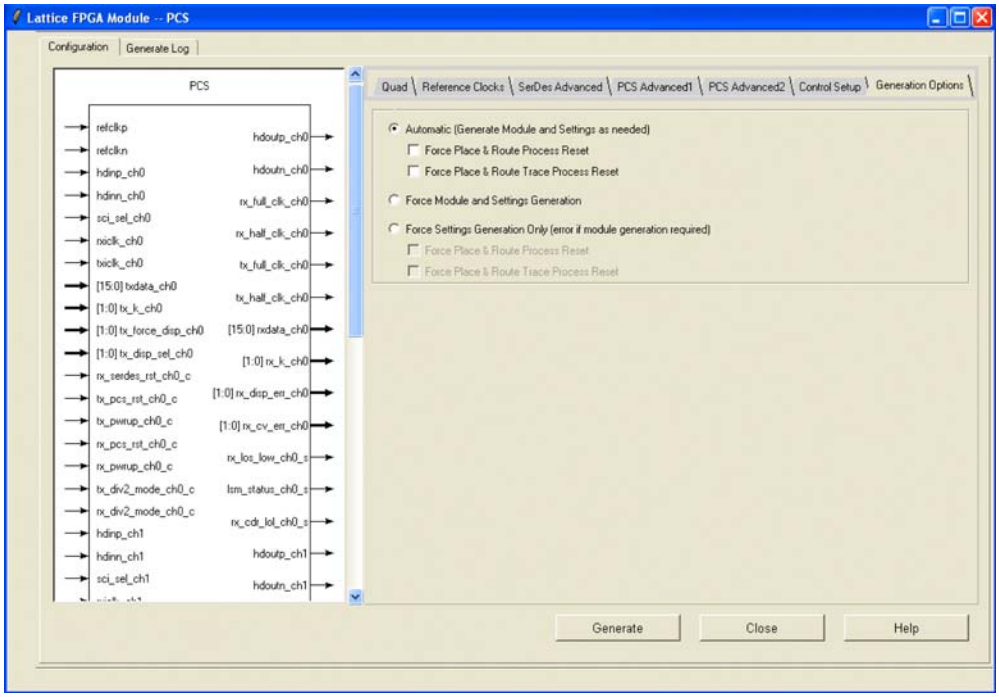


Table 10-18. SERDES_PCS GUI Attributes – Generation Options Tab

GUI Text	Description
Automatic	Automatically generates the HDL and configuration(.txt) files as needed. Some changes do not require regenerating both files.
Force Module and Settings Generation	Generates both the HDL and configuration files.
Force Settings Generation Only	Generates only the attributes file. You get an error message if the HDL file also needs to be generated.
Force Place & Route Process Reset	Resets the Place & Route Design process, forcing it to be run again with the newly generated PCS module.
Force Place & Route Trace Process Reset	Resets the Place & Route Trace process, forcing it to be run again with the newly generated PCS module.

Note:
Automatic is set as the default option. If either Automatic or Force Settings Generation Only and no sub-options (Process Reset Options) are checked and the HDL module is not generated, the reset pointer is set to Bitstream generation automatically.

After the Generation is finished, the reset marks in the process window will be reset accordingly.

Creating a New Simulation Project Using Simulation Wizard

This section describes how to use the Simulation Wizard to create a simulation project (.spf) file so you can import it into a standalone simulator.

1. In Project Navigator, click **Tools > Simulation Wizard**. The Simulation Wizard opens.
2. In the Preparing the Simulator Interface page click **Next**.
3. In the Simulator Project Name page, enter the name of your project in the Project Name text box and browse to the file path location where you want to put your simulation project using the Project Location text box and Browse button.

   When you designate a project name in this wizard page, a corresponding folder will be created in the file path you choose. Click **Yes** in the popup dialog that asks you if you wish to create a new folder.
4. Click either the Active-HDL® or ModelSim® simulator check box and click **Next**.
5. In the Process Stage page choose which type of Process Stage of simulation project you wish to create Valid types are RTL, Post-Synthesis Gate-Level, Post-Map Gate-Level, and Post-Route Gate-level+Timing. Only those process stages that are available are activated.

   Note that you can make a new selection for the current strategy if you have more than one defined in your project.

   The software supports multiple strategies per project implementation which allow you to experiment with alternative optimization options across a common set of source files. Since each strategy may have been processed to different stages, this dialog allows you to specify which stage you wish to load.
6. In the Add Source page, select from the source files listed in the Source Files list box or use the browse button on the right to choose another desired source file. Note that if you wish to keep the source files in the local simulation project directory you just created, check the **Copy Source to Simulation Directory** option.
7. Click **Next** and a Summary page appears and provides information on the project selections including the simulation libraries. By default, the Run Simulator check box is enabled and will launch the simulation tool you chose earlier in the wizard in the Simulator Project Name page.
8. Click **Finish**.

The Simulation Wizard Project (.spf) file and a simulation script DO file are generated after running the wizard. You can import the DO file into your current project if desired. If you are using Active-HDL, the wizard will generate an .ado file and if you are using ModelSim, it creates and .mdo file.

Note: PCS configuration file, (.txt) must be added in step 6.

Figure 10-39. Diamond Spreadsheet View (see Figure 10-5 for ispLEVER Equivalent)

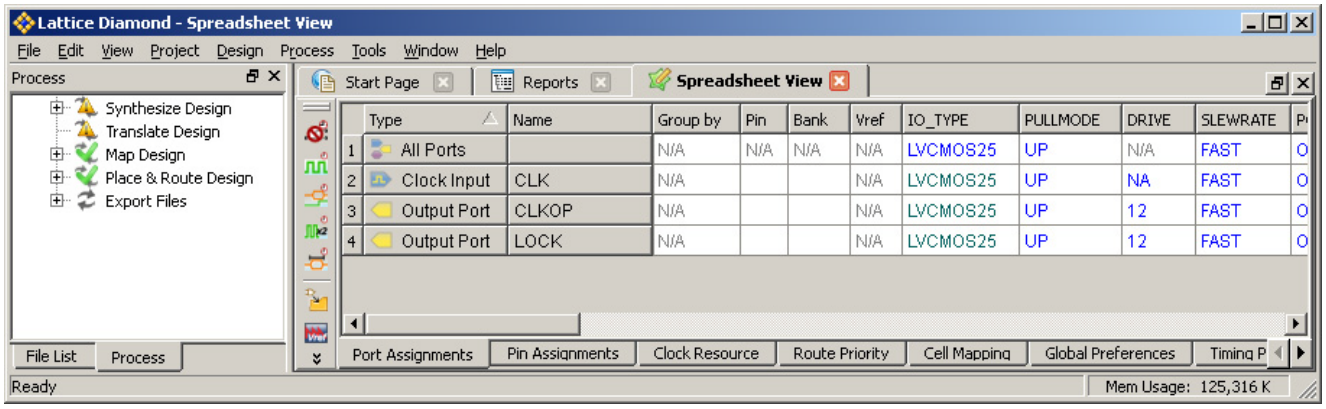


Figure 10-40. Diamond IPexpress Main Window (see Figure 10-14 for ispLEVER Equivalent)

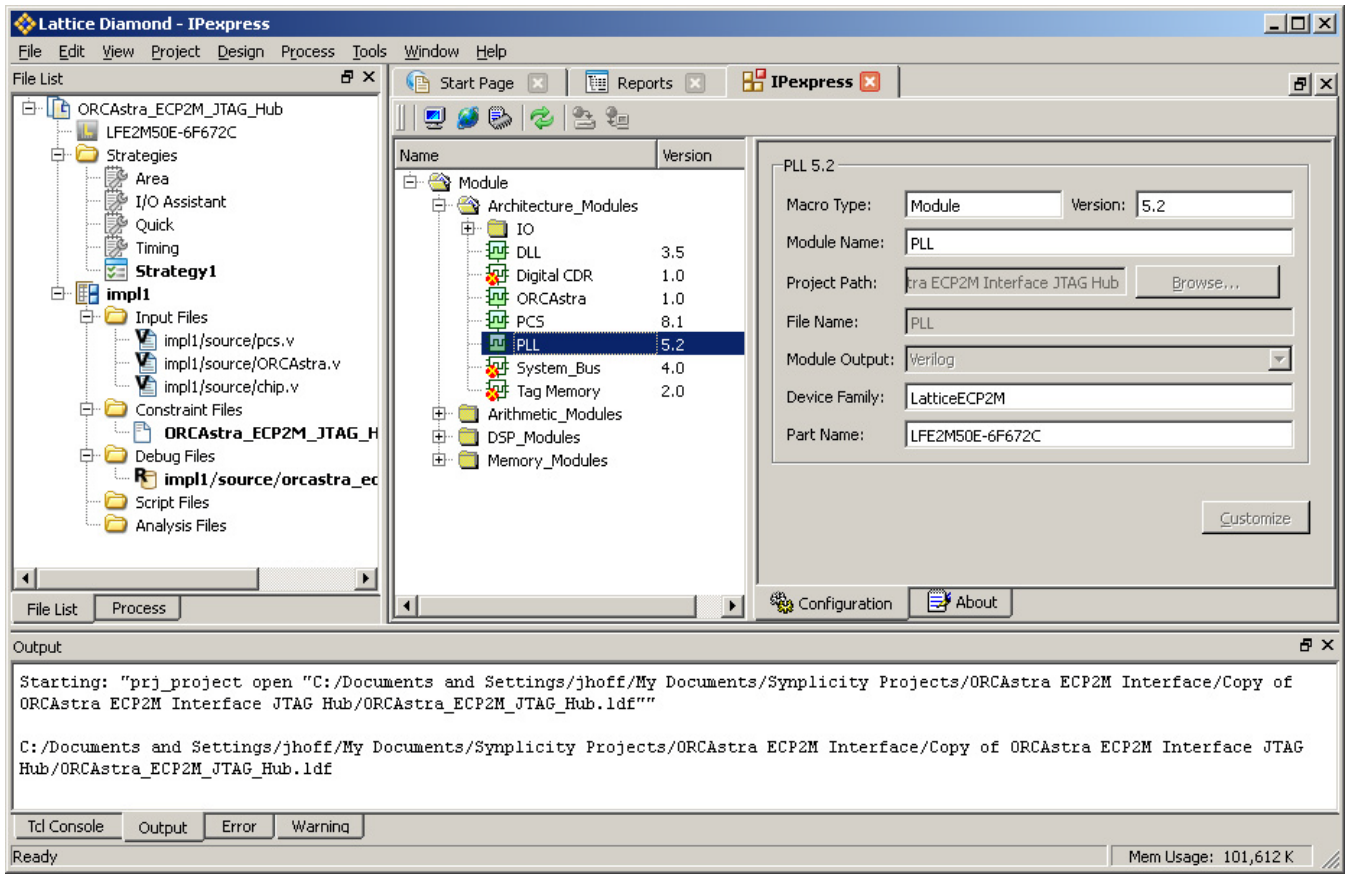


Figure 10-41. Diamond INPUT_SETUP Preference Window (see Figure 10-37 for ispLEVER Equivalent)