Lattice Radiant Software Guide for Lattice Diamond Users

May 9, 2020

Copyright

Trademarks

Disclaimers

Type Conventions Used in This Document

Convention	Meaning or Use
Bold	Items in the user interface that you select or click. Text that you type into the user interface.
<*Italic*>	Variables in commands, code syntax, and path names.
Ctrl+L	Press the two keys at the same time.
`Courier`	Code examples. Messages, reports, and prompts from the software.
`...`	Omitted material in a line of code.
`.` `.` `.`	Omitted lines in code and report examples.
`[ ]`	Optional items in syntax descriptions. In bus specifications, the brackets are required.
`( )`	Grouped items in syntax descriptions.
`{ }`	Repeatable items in syntax descriptions.
`|`	A choice between items in syntax descriptions.

Contents

Migrating Designs from Diamond to CrossLink-NX on the Radiant Software

When migrating ECP5 and CrossLink designs to CrossLink-NX on the Radiant™ software, you should expect to go through the normal design process, such as design entry, design analysis, debug, simulation, and testing. While Diamond and the Radiant software are very similar, there are substantial differences.

This chapter provides tips on how to rebuild your design using the Radiant IP, PMI, primitives, and constraints.

IP and Modules IP are basic, configurable modules that provide a variety of functions including I/O, arithmetic, memory, and more. The Radiant IP Catalog works similarly to the IP configuration in Diamond's Clarity Designer. IP Catalog offers a collection of IP that are similar to those found in Clarity Designer.

In the Radiant software, signal names of the generated components have been converted to lower case and "_i," "_o," and "_io" suffixes added. Some signals have been renamed. For example: DataA_Re to data_re_i and Cout to overflow_o.

For differences in specific IP, see:

- "Architecture IP" on page 9
- "Arithmetic IP" on page 15
- "DSP Arithmetic IP" on page 23
- "Memory IP" on page 31

For more information on the IP, see the Radiant Help under **References > Lattice Module Reference Guide**.

PMI PMI (Parameterized Module Instantiation) is an alternate way to use some of the components that come with IP Catalog. Instead of using IP

Catalog, PMI can directly instantiate a component into your HDL and customize it by setting parameters in the HDL. The Radiant software has a collection of PMI similar to Diamond's. To help you with PMI, templates for instantiating the modules are available in the Radiant Source Template view, which is similar to the Diamond Template Editor. See "PMI" on page 40.

Primitives Lattice library primitives are very basic functions, such as logic gates and flip-flops. Usually primitives are simply inferred in synthesis, but they can be directly instantiated as HDL into designs. See "Primitives" on page 45.

Preferences and Constraints Constraints are instructions applied to design elements that guide the design toward desired results and performance goals. The most common constraints are those for timing and pin assignments, but constraints are also available for placement, routing, and many other functions.

In Lattice Diamond, a Logical Preference File (.lpf) is used to constrain a design. In the Radiant software, preferences have been replaced by the industry standard Synopsys Design Constraints for ease of use and improved compatibility with third-party vendor tools such as Synopsys Synplify Pro.

This is one of the bigger differences between Diamond and Radiant designs. See "Constraints" on page 51.

Architecture IP

The Radiant software has IP similar to all of Diamond's architecture modules except for dynamic bank controller.

PLL

In Diamond for ECP5 and CrossLink, a PLL IP is generated using Clarity Designer. But in the Radiant software, IP Catalog is used to configure and generate the PLL IP, which can then be instantiated into the RTL.

The PLL IP is located under the Architecture_Modules folder. The tool allows you to provide an instance name, configure the PLL settings, and generate the PLL module.

Once the configuration is completed, the PLL module can be generated by clicking the Generate button. The generated module is inserted into the project in the form of an .ipx file. From there, you can find all necessary source, constraint, and configuration files generated from the tool. A right-click on the .ipx file allows you to create and copy a Verilog instantiation template or VHDL component declaration.

The following is the Verilog instantiation template from the generated PLL IP:

```
my_PLL u_PLL(.clki_i( ),
             .rstn_i( ),
             .clkop_o( ),
             .lock_o( )
            );
```

DDR Generic

There is a new data_coarse_dly_i signal.

Table 1: Feature Compatibility

Feature	CrossLink in Diamond: ddr_generic	ECP5 in Diamond: ddr_generic	CrossLink-NX in Radiant Software: DDR_Generic
Interface Type	Transmit, Receive	Transmit, Receive, Receive MIPI	Transmit, Receive
Enable Tri-state Control	Check box	Check box	Check box
I/O Standard for this Interface	Various	Various	Various
Bus Width for this interface	1-256	1-256	1-256

Table 1: Feature Compatibility (Continued)

Feature	CrossLink in Diamond: ddr_generic	ECP5 in Diamond: ddr_generic	CrossLink-NX in Radiant Software: DDR_Generic
Clock Frequency for this Interface	Depends on Interface Type, up to 600 MHz Transmit: 4.685 – 600 MHz Receive: 100 – 600 MHz	Depends on Interface Type, up to 400 MHz Transmit: 3.125 – 400 MHz Receive: 100 – 400 MHz Receive MIPI: 200 – 400 MHz	Depends on Gearing Ratio, up to 750 MHz X1: 100 – 250 MHz X2: 100 – 500 MHz X4: 100 – 750 MHz X5: 100 – 750 MHz
Clock to Data Relationship at the Pins	Edge-to-Edge, Centered	Depends on Interface Type Transmit and Receive: Edge-to-Edge, Centered Receive MIPI: Centered	Edge-to-Edge, Centered
Gearing Ratio	X1, X2	X1, X2	X1, X2, X4, X5
Organize Parallel Data	By Lane, By Time	By Lane, By Time	N/A
Enable ECLK Bridge	N/A	Check box	N/A
Data Path Delay	Depends on Interface Type Transmit: Bypass, Static User Defined, Dynamic User Defined Receive: Bypass, Static Default, Dynamic Default, Static User Defined, Dynamic User Defined	Depends on Interface Type Transmit: Bypass, Static User Defined, Dynamic User Defined Receive: Bypass, Static Default, Dynamic Default, Static User Defined, Dynamic User Defined Receive MIPI: Static Default, Static User Defined	Depends on Interface Type Transmit: Bypass, Static User Defined, Dynamic User Defined Receive: Bypass, Static Default, Dynamic Default, Static User Defined, Dynamic User Defined
Delay Value for User Defined	1-127	1-127	0-126 (Fine Delay for User Defined)
Coarse Delay Value for User Defined	N/A	N/A	0NS, 0P8NS, 1P6NS
Enable Dynamic Margin Control on Clock Delay	Check box	Check box	Fixed, Dynamic (Clock Path Delay)
Generate PLL with this Module	Check box	Check box	Check box (Enable PLL Instantiation)
PLL Input Clock Frequency	Various	Various	10 – (Clock Frequency for this interface) MHz
PLL Reference Clock From I/O Pin	Check box	Check box	Check box
CLKI Input Buffer Type	Various	Various	N/A
Reference Clock From I/O Pin	Check box	Check box	N/A

Table 1: Feature Compatibility (Continued)

Feature	CrossLink in Diamond: ddr_generic	ECP5 in Diamond: ddr_generic	CrossLink-NX in Radiant Software: DDR_Generic
Reference Clock Input Buffer Type	Various	Various	N/A
PLL Output Clock Tolerance	N/A	N/A	Percentage

GDDR 7:1

Table 2: Feature Compatibility

Feature	CrossLink in Diamond: gddr_7:1	ECP5 in Diamond: gddr_7:1	CrossLink-NX in Radiant Software: GDDR 7:1
Interface Type	Transmit, Receive	Transmit, Receive	Transmit, Receive
Bus Width	1-16	1-16	1-16
Interface Bandwidth	10 – 1200 Mbps	N/A	70-945 Mbps
Clock Frequency (Pixel Clock)	N/A	8-108 MHz	N/A
Enable Bit Alignment & Word Alignment Soft IP	Check box	Check box	Check box
Enable DELAYF Tuning (Uses Dynamic Input Delay block for adjusting Data and Clock Delay)	Check box	N/A	Check box (Enable Data Delay Control)
Reference Clock from I/O Pin	Check box	Check box	Check box
Reference Clock Input Buffer Type	Various	Various	Various

DDR MEM

Table 3: Feature Compatibility

Feature	ECP5 in Diamond: ddr_mem	CrossLink-NX in Radiant Software: DDR_MEM
Configuration (General) Tab		
Interface	DDR2, DDR3, DDR3L, LPDDR2, LPDDR3	DDR3, DDR3L, LPDDR2, LPDDR3
I/O Buffer Configuration	Various, depending on Interface	Various, depending on Interface
Gearing Ratio	N/A	X2, X4
DDR Memory Frequency	Various, depending on Interface	Various, depending on Interface

Table 3: Feature Compatibility (Continued)

Feature	ECP5 in Diamond: ddr_mem	CrossLink-NX in Radiant Software: DDR_MEM
DQS Buffer Configuration for DDR2	Single-ended, Differential	N/A
Number of DQ per DQS	4, 8	4, 8
Data Width	Various, depending on Interface	Various, depending on Interface
Data Mask	Check box	Check box
Clock/Address/Command	Check box	Check box
Enable Dynamic Margin Control on Clock Delay	Check box	Check box
Generate PLL with this module	Check box	N/A
PLL Input Clock Frequency	Various	N/A
PLL Reference Clock from I/O Pin	Check box	N/A
CLKI Input Buffer Type	Various	N/A
Clock/Address/Command Tab		
Number of Clocks	1, 2, 4	1, 2, 4
Address Width	13-16	13-16
Bank Address Width	2, 3	3
Number of Chip Selects	1, 2, 4	1, 2, 4
Advanced Settings		
DQS Read Delay Adjustment Enable	N/A	Check box
DQS Read Delay Adjustment	FACTORYONLY, PLUS, MINUS	POSITIVE, COMPLEMENT
DQS Read Delay Value	PLUS: 1-255; MINUS: 1-256	0-255
DQS Write Delay Adjustment Enable	N/A	Check box
DQS Write Delay Adjustment	FACTORYONLY, PLUS, MINUS	POSITIVE, COMPLEMENT
DQS Write Delay Value	PLUS: 1-255; MINUS: 1-256	0-255

MIPI_DPHY

Modified for CrossLink-NX:

- Differential data paths has been combined to a bus.
- Improved performance for the Hard MIPI implementation.

Added for CrossLink-NX:

- Hard CIL for MIPI DPHY protocol
- Support for HS reverse transmission

- Added LMMI and CIL ports
- Support for Soft MIPI_DPHY Transmitter
- Internal dedicated PLL for Hard MIPI DPHY

Table 4: Feature Compatibility

Feature	CrossLink in Diamond: mipi_dphy	CrossLink-NX in Radiant Software: MIPI_DPHY
Interface Type	Receive, Transmit	Receive, Transmit
MIPI Interface Application	CSI-2, DSI	CSI-2, DSI
DPHY Module Type	Hard MIPI DPHY (Tx/Rx), Soft MIPI DPHY (Rx only)	Hard MIPI DPHY (Tx/Rx), Soft MIPI DPHY (Tx/Rx)
MIXED DPHY Mode	N/A	CIL, No CIL (Crosslink Equivalent is No CIL)
DPHY PLL Mode	N/A	Internal, External
DPHY Clock Mode	N/A	Continuous, Non-Continuous
Interface Data Rate	20-1500 Mbps	80-2500 Mbps
Gearing Ratio	8:1, 16:1	8, 16
Bus Width	1-4	1-4
DPHY PLL Input Reference Frequency	24-200 MHz	24-200 MHz
Reference Clock from I/O pin	“supported”	“supported”
Reference Clock Input Buffer Type	“supported” – LVDS default	“supported” – MIPI DPHY default
Interface Clock Frequency	10-750 MHz	40-1250 MHz

SDR

CrossLink-NX has a new coarse_dly_i signal.

Table 5: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: sdr	CrossLink-NX in Radiant Software: SDR
Interface Type	Transmit, Receive	Transmit, Receive
Enable Tri-state Control	Check box	Check box
I/O Standard for this Interface	Various	Various
Bus Width for this interface	1-256	1-256
Clock Frequency for this Interface	1-200 MHz	1-300 MHz
Clock Inversion	Check box	Check box (Enable Clock Inversion)

Table 5: Feature Compatibility (Continued)

Feature	CrossLink and ECP5 in Diamond: sdr	CrossLink-NX in Radiant Software: SDR
Data Path Delay	Depends on Interface Type	Depends on Interface Type
	Transmit: Bypass, Static User Defined, Dynamic User Defined	Transmit: Bypass, Static User Defined, Dynamic User Defined
	Receive: Bypass, Static Default, Dynamic Default, Static User Defined, Dynamic User Defined	Receive: Bypass, Static Default, Dynamic Default, Static User Defined, Dynamic User Defined
Delay Value for User Defined	1-127	0-126 (Fine Delay)
Coarse Delay Value for User Defined	N/A	0NS, 0P8NS, 1P6NS

Arithmetic IP

The Radiant software has IP similar to all of Diamond's arithmetic modules except for fft_butterfly. The arithmetic IP of Diamond and the Radiant software are very similar except for a couple of differences:

- Data input widths often have a larger range. In Diamond, input widths are sometimes no more than 32 bits. In Radiant, input widths can be up to 64 bits.
- The Bus Ordering Style (Big Endian or Little Endian) option is not available in the Radiant software.

Adder

Table 6: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: adder	CrossLink-NX in Radiant Software: Adder
Specify the Data Width of the Adder	1-64	2-64 (Input Width)
Specify the Representation of the Adder	Signed, Unsigned	Signed, Unsigned (Input Signed)
Complex Inputs	Check box	Check box (Complex Number Addition)
Use Carry-in port	Check box	Check box (Carry-In Addition)
Specify the Carry-out Port	None, Carry-Out	None, Overflow (Carry-Out Addition)
Enable Output Register	Check box	Check box (Registered Outputs)
Specify number of pipeline stages	Depends on Data Width	Depends on Input Width (Pipelines)
Bus Ordering Style	Big Endian, Little Endian, None	N/A

Adder Subtractor

Table 7: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: adder_subtractor	CrossLink-NX in Radiant Software: Adder_Subtractor
Specify the Data Width of the Adder_Subtractor	1-64	2-64 (Data Width)
Specify the Representation of the Adder_Subtractor	Signed, Unsigned	Signed, Unsigned (Input Signed)
Complex Inputs	Check box	Check box (Inputs)
Use Carry-in port	Check box	Check box (Carry-In)

Table 7: Feature Compatibility (Continued)

Feature	CrossLink and ECP5 in Diamond: adder_subtractor	CrossLink-NX in Radiant Software: Adder_Subtractor
Specify the Carry-out Port	None, Carry-Out	None, Overflow (Carry-Out)
Enable Output Register	Check box	Check box (Enable Output Register)
Specify number of pipeline stages	Depends on Data Width	Depends on Input Width (Pipelines)
Bus Ordering Style	Big Endian, Little Endian, None	N/A

Comparator

Table 8: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: comparator	CrossLink-NX in Radiant Software: Comparator
Specify the data width of the comparator	1-64	2-64 (Data Width)
Specify the representation of comparator	Signed, Unsigned	Signed, Unsigned (Input Signed)
Specify the output port compare function	Various	Various (Compare Function)
Use LUT based implementation (to use lesser resources)	Check box	N/A
Enable Output Register	Check box	Check box (Enable Output Register)
Specify number of pipeline stages	Depends on data width	Depends on Data Width (Pipelines)
Bus Ordering Style	Big Endian, Little Endian, None	N/A

Complex Multiplier

Table 9: Feature Compatibility

Feature	CrossLink in Diamond: complex_multiplier	ECP5 in Diamond: complex_multiplier	CrossLink-NX in Radiant Software: Complex_Mult
Block Implementation	LUT	LUT, DSP	LUT, DSP (Implementation)
Input A Width	2-36	2-36	2-64 (Input A Width)
Input B Width	2-36	2-36	2-64 (Input B Width)
Representation	Signed, Unsigned	Signed, Unsigned	Signed, Unsigned (Input Signed)

Table 9: Feature Compatibility (Continued)

Feature	CrossLink in Diamond: complex_multiplier	ECP5 in Diamond: complex_multiplier	CrossLink-NX in Radiant Software: Complex_Mult
Specify the Number of Pipeline Stages	Depends on Input Width	Depends on Input Width and Block Implementation	Depends on Implementation (Pipelines)
Enable Input Registers	Check box	Check box	Check box (Registered Inputs)
Enable Output Registers	Check box	Check box	Check box (Registered Outputs)
Implementation	3 Multiplier, 4 Multiplier	3 Multiplier, 4 Multiplier	N/A
Bus Ordering Style	Big Endian, Little Endian, None	Big Endian, Little Endian, None	N/A

Convert

Table 10: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: convert	CrossLink-NX in Radiant Software: Convert
Input Width	1-256	1-256
Input Binary Point	0-7	0-7
Input Sign	Signed, Unsigned	SIGNED, UNSIGNED
Output Width	1-256	1-256
Output Binary Point	0-7	0-7
Output Sign	Signed, Unsigned	N/A
Rounding	Truncate, Nearest, Convergent	Truncate, Nearest, Convergent
Saturate	Wrap, Min_Max	Wrap, Min_Max

Counter

Table 11: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: counter	CrossLink-NX in Radiant Software: Counter
Specify the data width of the counter	1-64	1-64 (Data Width)
Specify the direction of the counter	Up, Down, Up-Down	Up, Down, UpDown (Counter Direction)
Optimized for speed	Check box	N/A

Table 11: Feature Compatibility (Continued)

Feature	CrossLink and ECP5 in Diamond: counter	CrossLink-NX in Radiant Software: Counter
Lower count value	Depends on data width	Depends on Data Width
Upper count value	Depends on data width	Depends on Data Width
Enable load input	Check box	Check box
Bus Ordering Style	Big Endian, Little Endian, None	N/A

LFSR

Table 12: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: lfsr	CrossLink-NX in Radiant Software: LFSR
LFSR Type	Fibonacci, Galois	Fibonacci, Galois
Gate Type	XOR, XNOR	XOR, XNOR
Number of Bits	1-512	1-512
Feedback Polynomial	Various	Various
Initial Value	Various	Various
Enable Parallel Output	Check box	Check box
Use Reloadable Seed Values	Check box	Check box
Bus Ordering Style	Big Endian, Little Endian, None	N/A

Multiply Accumulate

Table 13: Feature Compatibility

Feature	CrossLink in Diamond: multiply_accumulate	ECP5 in Diamond: multiply_accumulate	CrossLink-NX in Radiant Software: Mult_Accumulate
Block Implementation	LUT	LUT, DSP	LUT, DSP (Implementation)
Add/Sub Operation	Add, Sub	Add, Sub	Addition, Subtraction (Addition/Subtraction)
Input A Width	2-36	2-36	2-64
Representation	Signed, Unsigned	Signed, Unsigned	Signed, Unsigned
Input B Width	2-36	2-36	2-64
Representation	Signed, Unsigned	Signed, Unsigned	Signed, Unsigned

Table 13: Feature Compatibility (Continued)

Feature	CrossLink in Diamond: multiply_accumulate	ECP5 in Diamond: multiply_accumulate	CrossLink-NX in Radiant Software: Mult_Accumulate
Accumulator Width	1-32	1-32	Depends on Input Width
Specify the Number of Pipeline Stages	Depends on Input Width	Depends on Input Width and Block Implementation	Depends on Implementation (Pipelines)
Enable Input Registers	Check box	Check box	Check box (Registered Inputs)
Enable Output Registers	Check box	Check box	Check box (Registered Outputs)
Bus Ordering Style	Big Endian, Little Endian, None	Big Endian, Little Endian, None	N/A

Multiply Add Subtract

Table 14: Feature Compatibility

Feature	CrossLink in Diamond: mult_add_sub	ECP5 in Diamond: mult_add_sub	CrossLink-NX in Radiant Software: Mult_Add_Sub
Block Implementation	LUT	LUT, DSP	LUT, DSP (Implementation)
Add/Sub Operation	Add, Sub	Add, Sub	Addition, Subtraction (Addition/Subtraction)
Input A0/A1 Width	2-36	2-36	2-64 (Input A Width)
Representation	Signed, Unsigned	Signed, Unsigned	Check box (Input A Signed)
Input B0/B1 Width	2-36	2-36	2-64 (Input B Width)
Representation	Signed, Unsigned	Signed, Unsigned	Check box (Input B Signed)
Specify the Number of Pipeline Stages	Depends on Input Width	Depends on Input Width and Block Implementation	Depends on Implementation (Pipelines)
Enable Input Registers	Check box	Check box	Check box (Registered Inputs)
Enable Output Registers	Check box	Check box	Check box (Registered Outputs)
Bus Ordering Style	Big Endian, Little Endian, None	Big Endian, Little Endian, None	N/A

Multiply Add Subtract Sum

Table 15: Feature Compatibility

Feature	CrossLink in Diamond: mult_add_sub_sum	ECP5 in Diamond: mult_add_sub_sum	CrossLink-NX in Radiant Software: Mult_Add_Sub_Sum
Block Implementation	LUT	LUT, DSP	LUT, DSP (Implementation)
Add/Sub 0 Operation	Add, Sub	Add, Sub	Add, Sub
Add/Sub 1 Operation	Add, Sub	Add, Sub	Add, Sub
Input A0/A1/A2/A3 Width	2-36	2-36	2-64
Representation	Signed, Unsigned	Signed, Unsigned	Signed, Unsigned
Input B0/B1/B2/B3 Width	2-36	2-36	2-64
Representation	Signed, Unsigned	Signed, Unsigned	Signed, Unsigned
Specify the Number of Pipeline Stages	Depends on Input Width	Depends on Input Width and Block Implementation	Depends on Input Width (Pipelines)
Enable Input Registers	Check box	Check box	Check box (Registered Inputs)
Enable Output Registers	Check box	Check box	Check box (Registered Outputs)
Bus Ordering Style	Big Endian, Little Endian, None	Big Endian, Little Endian, None	N/A

Multiplier

Table 16: Feature Compatibility

Feature	CrossLink in Diamond: multiplier	ECP5 in Diamond: multiplier	CrossLink-NX in Radiant Software: Multiplier
Use a Constant Coefficient	Check box	Check box	Check box (Use Multiplication Co-efficient)
Constant Coefficient Value	(-2^31) to (2^31-1)	(-2^31) to (2^31-1)	(-2^31) to (2^31-1)
Use RAM Based Multiplier	Check box	Check box	N/A
Block Implementation	LUT	LUT, DSP	LUT, DSP (Implementation)
Input A Width	2-36	2-36	2-64
Representation	Signed, Unsigned	Signed, Unsigned	Signed, Unsigned (Input A Signed)
Input B Width	2-36	2-36	2-64

Table 16: Feature Compatibility (Continued)

Feature	CrossLink in Diamond: multiplier	ECP5 in Diamond: multiplier	CrossLink-NX in Radiant Software: Multiplier
Representation	Signed, Unsigned	Signed, Unsigned	Signed, Unsigned (Input B Signed)
Specify the Number of Pipeline Stages	Depends on Input Width	Depends on Input Width and Block Implementation	Depends on Block Implementation (Pipelines)
Enable Input Registers	Check box	Check box	Check box (Registered Inputs)
Enable Output Registers	Check box	Check box	Check box (Registered Outputs)
Bus Ordering Style	Big Endian, Little Endian, None	Big Endian, Little Endian, None	N/A

Sin-Cos Table

Table 17: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: sin-cos_table	CrossLink-NX in Radiant Software: Sin_Cos_Table
Block Implementation	LUT, EBR	LUT, EBR
Input Theta Bit Width	3-10	3-10
Output Result Bit Width	4-32	4-32
Output Mode	Sin, Cos, Sin-Cos	Sin, Cos, Sin-Cos
Use Tables for Quarter-wave only	Check box	Check box
Use 1-bit for Signed Integer	Check box	Check box
Enable Input Registers	Check box	Check box
Enable Output Registers	Check box	Check box
Specify the number of pipeline stages	1-3	1-3
Bus Ordering Style	Big Endian, Little Endian, None	N/A

Subtract

Table 18: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: subtractor	CrossLink-NX in Radiant Software: Subtractor
Specify the Data Width of the Subtractor	1-64	2-64 (Input Width)
Specify the Representation of the Subtractor	Signed, Unsigned	Signed, Unsigned (Input Signed)
Complex Inputs	Check box	Check box (Complex Number Subtraction)
Use Carry-in port	Check box	Check box (Carry-In Subtraction)
Specify the Carry-out Port	None, Carry-Out	None, Overflow (Carry-Out Subtraction)
Enable Output Register	Check box	Check box (Registered Outputs)
Specify number of pipeline stages	Depends on Data Width	Depends on Input Width (Pipelines)
Bus Ordering Style	Big Endian, Little Endian, None	N/A

DSP Arithmetic IP

For ECP5, Diamond provides several IP that implement arithmetic functions using DSP blocks. For CrossLink-NX, the Radiant software has some that are similar to the IP for ECP5 in Diamond but with fewer options to make configuration simpler. To see the differences, use the following tables. The DSP arithmetic IP for CrossLink-NX are:

- DSP_Mult_Accumulate
- DSP_Mult_Add_Sub
- DSP_Mult_Add_Sub_Sum
- DSP_Multiplier

DSP Multiply Accumulate

Table 19: Feature Compatibility

Feature	ECP5 in Diamond: mac	CrossLink-NX in Radiant Software: DSP_Mult_Accumulate
Input A Width	2-72	2-72
Input B Width	2-72	2-72
Data Type A	Signed, Unsigned, Dynamic	Signed, Unsigned
Data Type B	Signed, Unsigned, Dynamic	Signed, Unsigned
Source A	Parallel, Shift, Dynamic	N/A
Source B	Parallel, Shift, Dynamic	N/A
Select Shift Out A	Check box	N/A
Select Shift Out B	Check box	N/A
Add/Sub Operation	Add, Sub, Dynamic	N/A [4]
Input Register A	Check box	Check box
Input A Clock	CLK0, CLK1	N/A [1]
Input A Clock Enable	CE0, CE1, CE2, CE3	N/A [2]
Input A Reset	RST0, RST1	N/A [3]
Input Register B	Check box	Check box
Input B Clock	CLK0, CLK1	N/A [1]
Input B Clock Enable	CE0, CE1, CE2, CE3	N/A [2]
Input B Reset	RST0, RST1	N/A [3]
Pipeline Mode	Check box	N/A
Pipeline Register	Check box	Check box

Table 19: Feature Compatibility (Continued)

Feature	ECP5 in Diamond: mac	CrossLink-NX in Radiant Software: DSP_Mult_Accumulate
Output Register	Check box	Check box
Bus Ordering Style	Big Endian, Little Endian, None	N/A

[1] Two clocks were used in ECP5. However, CrossLink-NX DSP primitives only use a single clock for input and output.

[2] ECP5 has four selectable clock enables (CE0, CE1, CE2, and CE3). In CrossLink-NX DSPs, the clock enables are already designated (ce_a_i for input A, ce_b_i for input B, ce_p_i for pipeline, and ce_o_i for output).

[3] Two resets are used in ECP5. However CrossLink-NX uses designated resets for input and output (rst_a_i for input A, rst_b_i for input B, rst_p_i for pipeline, and rst_o _i for output).

[4] The Add/Sub operation in the Radiant software is always dynamic.

Table 20: Port Compatibility

ECP5 in Diamond: mac	CrossLink-NX in Radiant Software: DSP_Mult_Accumulate
CLK0	N/A
CLK1	N/A
N/A	clk_i
CE0	N/A
CE1	N/A
CE2	N/A
CE3	N/A
N/A	ce_a_i
N/A	ce_b_i
N/A	ce_o_i
RST0	N/A
RST1	N/A
N/A	rst_a_i
N/A	rst_b_i
N/A	rst_o_i
SignA	N/A
SignB	N/A
SourceA	N/A

Table 20: Port Compatibility (Continued)

ECP5 in Diamond: mac	CrossLink-NX in Radiant Software: DSP_Mult_Accumulate
SourceB	N/A
ACCUMSLOAD	accumsload
ADDNSUB	addnsub_i
A	input_a_i
B	input_b_i
LD	ld
SRIA	N/A
SRIB	N/A
ACCUM	result_o
SROA	N/A
SROB	N/A
OVERFLOW	N/A

DSP Multiply Add Subtract

Table 21: Feature Compatibility

Feature	ECP5 in Diamond: multaddsub	CrossLink-NX in Radiant Software: DSP_Mult_Add_Sub
Enable Cascade Input	Check box	N/A
Input A0/A1 Width	2-72	2-72
Input B0/B1 Width	2-72	2-72
Data Type A	Signed, Unsigned, Dynamic	Signed, Unsigned
Data Type B	Signed, Unsigned, Dynamic	Signed, Unsigned
Reset Mode	SYNC, ASYNC	SYNC, ASYNC
Input A0 Source	Parallel, Shift, Dynamic	N/A
Input A1 Source	Parallel, Shift, Dynamic	N/A
Input B0 Source	Parallel, Shift, Dynamic	N/A
Input B1 Source	Parallel, Shift, Dynamic	N/A
Select Shift Out A	Check box	N/A
Cascade Match Register	Check box	N/A
Select Shift Out B	Check box	N/A
Add/Sub Operation	Add, Sub, Dynamic	N/A [4]

Table 21: Feature Compatibility (Continued)

Feature	ECP5 in Diamond: multaddsub	CrossLink-NX in Radiant Software: DSP_Mult_Add_Sub
Input Register A	Check box	N/A
Input A Clock	CLK0, CLK1	N/A [1]
Input A Clock Enable	CE0, CE1, CE2, CE3	N/A [2]
Input A Reset	RST0, RST1	N/A [3]
Input Register B	Check box	N/A
Input B Clock	CLK0, CLK1	N/A [1]
Input B Clock Enable	CE0, CE1, CE2, CE3	N/A [2]
Input B Reset	RST0, RST1	N/A [3]
Input Register AB0	N/A	Check box
Input Register AB1	N/A	Check box
Pipeline Mode	Check box	N/A
Enable Pipeline Register	Check box	Check box
Output Register	Check box	Check box
Output Clock	InputA, InputB	N/A
Input Output Clock Enable	CE0, CE1, CE2, CE3	N/A [2]
Bus Ordering Style	Big Endian, Little Endian, None	N/A

[1] Two clocks were used in ECP5. However, CrossLink-NX DSP primitives only use a single clock for input and output.

[2] ECP5 has four selectable clock enables (CE0, CE1, CE2, and CE3). In CrossLink-NX DSPs, the clock enables are already designated (ce_a_i for input A, ce_b_i for input B, ce_p_i for pipeline, and ce_o_i for output).

[3] Two resets are used in ECP5. However CrossLink-NX uses designated resets for input and output (rst_a_i for input A, rst_b_i for input B, rst_p_i for pipeline, and rst_o _i for output).

[4] The Add/Sub operation in the Radiant software is always dynamic.

Table 22: Port Compatibility

ECP5 in Diamond: multaddsub	CrossLink-NX in Radiant Software: DSP_Mult_Add_Sub
CLK0	N/A
CLK1	N/A
N/A	clk_i
CE0	N/A
CE1	N/A

Table 22: Port Compatibility (Continued)

ECP5 in Diamond: multaddsub	CrossLink-NX in Radiant Software: DSP_Mult_Add_Sub
CE2	N/A
CE3	N/A
N/A	ce_a_i
N/A	ce_b_i
N/A	ce_p_i
N/A	ce_o_i
RST0	N/A
RST1	N/A
N/A	rst_a_i
N/A	rst_b_i
N/A	rst_p_i
N/A	rst_o_i
SignA	N/A
SignB	N/A
ShiftA0	N/A
ShiftA1	N/A
ShiftB0	N/A
ShiftB1	N/A
SourceA	N/A
SourceB	N/A
ADDNSUB	addnsub_i
A0	input_0_i
A1	input_a1_i
B0	input_b0_i
B1	input_b1_i
SRIA	N/A
SRIB	N/A
CIN	N/A
SignCIN	N/A
SignSUM	N/A
SROA	N/A

Table 22: Port Compatibility (Continued)

ECP5 in Diamond: multaddsub	CrossLink-NX in Radiant Software: DSP_Mult_Add_Sub
SROB	N/A
SUM	result_o

DSP Multiply Add Subtract Sum

Added a designated clock enable and reset for input and output.

Removed for CrossLink-NX:

- Select shift Out A and Select shift Out B.
- Add/Sub 0 Operation (Add, Sub, or Dynamic).
- Add/Sub 1 Operation (Add, Sub, or Dynamic).
- Input Source A0, A1, A2, A3, B0, B1, B2, and B3 (Parallel, Shift, or Dynamic).
- Bus Ordering style not supported by primitive.
- Dynamic mode removed from Data Type.
- Ports:
  - SROA (Shift Output)
  - SROB (Shift Output)
  - OVERFLOW
  - SourceA
  - SourceB
  - SignA
  - SignB
  - SRIA
  - SRIB (Shift Input)

Modified for CrossLink-NX:

- Multiple clock support to single clock. Special primitive only uses single clock.

DSP Multiplier

Table 23: Feature Compatibility

Feature	ECP5 in Diamond: mult	CrossLink-NX in Radiant Software: DSP_Multiplier
Input Width	2-72	2-72
Data Type	Signed, Unsigned, Dynamic	Signed, Unsigned
Source	Parallel, Shift, Dynamic	N/A
Select Shift Out A	Check box	N/A
Select Shift Out B	Check box	N/A
Input Register A	Check box	Check box
Input A Clock	CLK0, CLK1	N/A [1]
Input A Clock Enable	CE0, CE1, CE2, CE3	N/A [2]
Input A Reset	RST0, RST1	N/A [3]
Input Register B	Check box	Check box
Input B Clock	CLK0, CLK1	N/A [1]
Input B Clock Enable	CE0, CE1, CE2, CE3	N/A [2]
Input B Reset	RST0, RST1	N/A [3]
Pipeline Mode	Check box	N/A
Pipeline Register	Check box	N/A
Output Register	Check box	Check box
Bus Ordering Style	Big Endian, Little Endian, None	N/A

[1] Two clocks were used in ECP5. However, CrossLink-NX DSP primitives only use a single clock for input and output.

[2] ECP5 has four selectable clock enables (CE0, CE1, CE2, and CE3). In CrossLink-NX DSPs, the clock enables are already designated (ce_a_i for input A, ce_b_i for input B, and ce_o_i for output).

[3] Two resets are used in ECP5. However CrossLink-NX uses designated resets for input and output (rst_a_i for input A, rst_b_i for input B, and rst_o _i for output).

Table 24: Port Compatibility

ECP5 in Diamond: mult	CrossLink-NX in Radiant Software: DSP_Multiplier
CLK0	N/A
CLK1	N/A
N/A	clk_i
CE0	N/A

Table 24: Port Compatibility (Continued)

ECP5 in Diamond: mult	CrossLink-NX in Radiant Software: DSP_Multiplier
CE1	N/A
CE2	N/A
CE3	N/A
N/A	ce_a_i
N/A	ce_b_i
N/A	ce_o_i
RST0	N/A
RST1	N/A
N/A	rst_a_i
N/A	rst_b_i
N/A	rst_o_i
SignA	N/A
SignB	N/A
SourceA	N/A
SourceB	N/A
A	input_a_i
B	input_b_i
SRIA	N/A
SRIB	N/A
P	result_o
SROA	N/A
SROB	N/A

Memory IP

The types of memory available in the Radiant software are very similar to what is available in Diamond. All the IP available in Diamond have equivalents in the Radiant IP Catalog. Design differences are just in the names and a few options.

If a memory initialization file is needed, create one before configuring the IP. Each row includes the value to be stored in a particular memory location. The file name for the memory initialization file is *.mem (<*file_name*>.mem).

FIFO

Removed for CrossLink-NX:

- ECC
- ERROR port

Table 25: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: fifo	CrossLink-NX in Radiant Software: FIFO
FIFO Implementation	EBR Only, LUT Only	EBR, LUT (Implementation Type)
Address Depth	2-65536	2-65536
Data Width	1-256	1-256
Total Memory bits	N/A	Calculated value from Address Depth and Data width
Enable Output Register	Check box	Check box
Controlled by RdEn	Check box	N/A
Flag Control: Almost Empty Flag	Check box	Check box (Enable ALMOST EMPTY flag)
Flag Control:	Static - Single Threshold, Static - Dual Threshold, Dynamic - Single Threshold, Dynamic - Dual Threshold	Static - Single Threshold, Static - Dual Threshold, Dynamic - Single Threshold, Dynamic - Dual Threshold (ALMOST EMPTY assertion type)
Flag Control: Assert	1-512	1-1023 (Assert LEVEL)
Flag Control: Deassert	1-512	2-1022 (Deassert LEVEL)
Flag Control: Almost Full Flag	Check box	Check box (Enable ALMOST FULL assertion flag)
Flag Control	Static - Single Threshold, Static - Dual Threshold, Dynamic - Single Threshold, Dynamic - Dual Threshold	Static - Single Threshold, Static - Dual Threshold, Dynamic - Single Threshold, Dynamic - Dual Threshold (ALMOST FULL assertion type)

Table 25: Feature Compatibility (Continued)

Feature	CrossLink and ECP5 in Diamond: fifo	CrossLink-NX in Radiant Software: FIFO
Flag Control: Assert	1-512	2-1023 (Assert LEVEL)
Flag Control: Deassert	1-512	1-1022 (Deassert LEVEL)
Reset Mode: Assertion	Async, Sync	N/A
Reset Mode: Release	Async, Sync	N/A
Reset Assertion	N/A	sync, async
Data Count (Synchronized with Write Clock)	Check box	Check box (Enable Data Count)
Enable ECC (not supported for Data Width > 64)	Check box	N/A
Bus Ordering Style	Big Endian, Little Endian, None	N/A

FIFO DC

Removed for CrossLink-NX:

- ECC
- ERROR port

Table 26: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: fifo_dc	CrossLink-NX in Radiant Software: FIFO_DC
FIFO Implementation	EBR Only, LUT Only	EBR, LUT (Implementation Type)
Write Address Depth	2-65536	2-65536 (Write Port: Address Depth)
Data Width	1-256	1-256 (Write Port Data Width)
Read Address Depth	2-65536	2-65536 (Read Port: Address Depth)
Data Width	1-256	1-256 (Read Port: Data Width)
Total Memory bits	N/A	Calculated value from Address Depth and Data width
Enable Output Register	Check box	Check box
Controlled by RdEn	Check box	N/A
Flag Control: Almost Empty Flag	Check box	Check box (Enable ALMOST EMPTY flag)
Flag Control	Static - Single Threshold, Static - Dual Threshold, Dynamic - Single Threshold, Dynamic - Dual Threshold	Static - Single Threshold, Static - Dual Threshold, Dynamic - Single Threshold, Dynamic - Dual Threshold (ALMOST EMPTY assertion type)

Table 26: Feature Compatibility (Continued)

Feature	CrossLink and ECP5 in Diamond: fifo_dc	CrossLink-NX in Radiant Software: FIFO_DC
Flag Control: Assert	1-512	1-511 (Assert LEVEL)
Flag Control: Deassert	1-512	2-511 (Deassert LEVEL)
Flag Control: Almost Full Flag	Check box	Check box (Enable ALMOST FULL flag)
Flag Control	Static - Single Threshold, Static - Dual Threshold, Dynamic - Single Threshold, Dynamic - Dual Threshold	Static - Single Threshold, Static - Dual Threshold, Dynamic - Single Threshold, Dynamic - Dual Threshold (ALMOST FULL assertion type)
Flag Control: Assert	1-512	1-511 (Assert LEVEL)
Flag Control: Deassert	1-512	1-510 (Deassert LEVEL)
Reset Mode: Assetion	Async, Sync	N/A
Reset Mode: Release	Async, Sync	N/A
Reset Assertion	N/A	sync, async
Data Count (Synchronized with Write Clock)	Check box	Check box (Enable Data Count (Write))
Data Count (Synchronized with Read Clock)	Check box	Check box (Enable Data Count (Read))
Enable ECC (not supported for Data Width > 64)	Check box	N/A
Bus Ordering Style	Big Endian, Little Endian, None	N/A

RAM-Based Shift Register

Removed for CrossLink-NX:

- Variable (lossy) configuration
- Memory initialization

Table 27: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: ram-based_shift_register	CrossLink-NX in Radiant Software: Shift_Register
Max Shift	N/A	2-8192
Data Width	1 to 256	1-256
Ram Type	LUT Based, EBR Based	EBR, LUT (Implementation Type)
Shift Register Type	Fixed Length, Variable Length (Lossy), Variable Length (Lossless)	fixed, variable

Table 27: Feature Compatibility (Continued)

Feature	CrossLink and ECP5 in Diamond: ram-based_shift_register	CrossLink-NX in Radiant Software: Shift_Register
Shift Register Type: Fixed Length	2-1024	N/A
Shift Register Type: Variable Length (Lossy)	2-1024	N/A
Shift Register Type: Variable Length (Lossless)	2-1024	N/A
Total Memory bits	N/A	Calculated value from Address Depth and Data width
Enable Output Register	Check box	Check box
Memory File	User input	N/A
Memory Initialization	N/A	N/A
Memory File Format	Binary, Hex, Addressed Hex	N/A
Bus Ordering Style	Big Endian, Little Endian, None	N/A

Distributed DPRAM

Added rd_clock port. However, you must still handle clock crossing between addresses.

Table 28: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: distributed_dpram	CrossLink-NX in Radiant Software: Distributed_DPRAM
Address Depth	2-8192	2-32768 (Write Port: Address Depth)
Data Width	1-256	1-512 (Write Port: Data Width)
Read Port: Address Depth	N/A	Same value as Write Port: Address Depth
Read Port: Data Width	N/A	Same value as Write Port: Data Width
Enable Output Register	Check box	Check box (Read Port: Enable Output Register)
Reset Assertion	N/A	sync
Memory File	User input	User input
Memory Initialization	N/A	none, Initialize all to 0s, Initialize all to 1s, Memory File
Memory File Format	Binary, Hex, Addressed Hex	hex, binary
Bus Ordering Style	Big Endian, Little Endian, None	N/A

Distributed SPRAM

Table 29: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: distributed_spram	CrossLink-NX in Radiant Software: Distributed_SPRAM
Address Depth	2-8192	2-8192
Data Width	1-256	1-256
Total Memory bits	N/A	Calculated value from Address Depth and Data width
Enable Output Register	Check box	Check box
Reset Assertion	N/A	sync, async
Memory File	User input	User input
Memory Initialization	N/A	none, Initialize all to 0s, Initialize all to 1s, Memory File
Memory File Format	Binary, Hex, Addressed Hex	hex, binary
Bus Ordering Style	Big Endian, Little Endian, None	N/A

Distributed ROM

Table 30: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: distributed_rom	CrossLink-NX in Radiant Software: Distributed_ROM
Address Depth	2-8192	2-65536 (Read Port: Address Depth)
Data Width	1-128	1-512 (Read Port: Data Width)
Total Memory bits	N/A	Calculated value from Address Depth and Data width
Enable Output Register	Check box	Check box (Read Port: Enable Output Register)
Reset Assertion	N/A	sync, async
Memory File	User input	User input
Memory Initialization	N/A	none, Initialize all to 0s, Initialize all to 1s, Memory File
Memory File Format	Binary, Hex, Addressed Hex	hex, binary
Bus Ordering Style	Big Endian, Little Endian, None	N/A

RAM DP

Removed for CrossLink-NX:

- ECC
- ERROR port

Table 31: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: ram_dp	CrossLink-NX in Radiant Software: RAM_DP
Read Port		
Address Depth	2-65536	2-65536 (Read Port: Address Depth)
Data Width	1-256	1-256 (Read Port: Address Depth)
Write Port		
Address Depth	2-65536	2-65536 (Read Port: Address Depth)
Data Width	1-256	1-256 (Read Port: Address Depth)
Total Memory bits	N/A	Calculated value from Address Depth and Data width
Provide Byte Enables	Check box	N/A
Byte Size	8, 9	N/A
Enable Output Register	Check box	Check box
Enable Output ClockEn	Check box	Check box
Optimization	Area, Speed	N/A
Reset Mode	Async, Sync	sync, async
Release	Async, Sync	N/A
Enable Byte Enable	N/A	Check box
Initialization	Initialize to all 0's, Initialize to all 1's, Memory File	none, Initialize all to 0s, Initialize all to 1s, Memory File
Memory File Format	Binary, Hex, Addressed Hex	hex, binary
Enable ECC (not supported for Data Width > 64)	Check box	Check box (Enable ECC)
Pipeline Stages for Q and ERROR Outputs	0-2	N/A
Bus Ordering Style	Big Endian, Little Endian, None	N/A

RAM DP True

Removed for CrossLink-NX:

- ECC

- Read-Before-Write
- Write through
- ERROR port

Table 32: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: ram_dp_true	CrossLink-NX in Radiant Software: RAM_DP_True
A Port: Address Depth	2-65536	2-65536 (Read Port: Address Depth)
A Port: Data Width	1-256	1-256 (Read Port: Address Depth)
A Port: Enable Output Register	Check box	Check box (Read Port: Enable Output Register (A)
A Port: Enable Output ClockEn	Check box	N/A
B Port: Address Depth	2-65536	2-65536 (Read Port: Address Depth)
B Port: Data Width	1-256	1-256 (Read Port: Address Depth)
B Port: Enable Output Register	Check box	Check box (Read Port: Enable Output Register (B)
B Port: Enable Output ClockEn	Check box	N/A
Total Memory bits	N/A	Calculated value from Address Depth and Data width
Provide Byte Enables	Check box	N/A
Byte Size	8, 9	N/A
Optimization	Area, Speed	N/A
Reset Mode	Async, Sync	N/A
Reset Assertion (A)	N/A	sync, async
Reset DeAssertion (A)	N/A	sync, async
Reset Assertion (B)	N/A	sync, async
Reset DeAssertion (B)	N/A	sync, async
Release	Async, Sync	N/A
Enable Byte Enable (A)	N/A	Check box
Enable Byte Enable (B)	N/A	Check box
Initialization	Initialize to all 0’s, Initialize to all 1’s, Memory File	none, Initialize all to 0s, Initialize all to 1s, Memory File
Memory File Format	Binary, Hex, Addressed Hex	hex, binary
Enable ECC (not supported for Data Width > 64)	Check box	Check box (Enable ECC)
Pipeline Stages for Q and ERROR Outputs	0-2	N/A
Bus Ordering Style	Big Endian, Little Endian, None	N/A

Table 32: Feature Compatibility (Continued)

Feature	CrossLink and ECP5 in Diamond: ram_dp_true	CrossLink-NX in Radiant Software: RAM_DP_True
Port A Write Mode	Normal, Write Through, Read before Write	N/A
Port B Write Mode	Normal, Write Through, Read before Write	N/A

RAM DQ

Removed for CrossLink-NX:

- ECC
- Read-Before-Write
- Write through
- ERROR port

Modified for CrossLink-NX:

- Byte-Enable is supported, however the Radiant software automatically applies the byte-width. Nine is used when data_width is divisible by 9, and 8 is used when data_width is divisible by 8, with 9 being priority. So for configurations divisible with both (for example, DWID = 72), the byte-width is 9.

Table 33: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: ram_dq	CrossLink-NX in Radiant Software: RAM_DQ
Address Depth	2-65536	2-65536
Data Width	1-256	1-512
Total Memory bits	N/A	Calculated value from Address Depth and Data width
Provide Byte Enables	Check box	N/A
Byte Size	8, 9	N/A
Enable Output Register	Check box	Check box
Enable Output ClockEn	Check box	Check box
Optimization	Area, Speed	N/A
Reset Mode	Async, Sync	sync, async
Release	Async, Sync	N/A
Enable Byte Enable	N/A	Check box
Initialization	Initialize to all 0’s, Initialize to all 1’s, Memory File	none, Initialize all to 0s, Initialize all to 1s, Memory File

Table 33: Feature Compatibility (Continued)

Feature	CrossLink and ECP5 in Diamond: ram_dq	CrossLink-NX in Radiant Software: RAM_DQ
Memory File Format	Binary, Hex, Addressed Hex	hex, binary
Enable ECC (not supported for Data Width > 64)	Pipeline Stages for Q and ERROR Outputs (0, 1, 2)	N/A
Bus Ordering Style	Big Endian, Little Endian, None	N/A

ROM

Removed for CrossLink-NX:

- ECC
- ERROR port

Table 34: Feature Compatibility

Feature	CrossLink and ECP5 in Diamond: rom	CrossLink-NX in Radiant Software: ROM
Address Depth	2-65536	2-65536
Data Width	1-256	1-512
Enable Output Register	Check box	Check box
Enable Output ClockEn	Check box	Check box
Total Memory bits	N/A	Calculated value from Address Depth and Data width
Optimization	Area, Speed	N/A
Reset Mode : Assetion	Async, Sync	N/A
Reset Mode : Release	Async, Sync	N/A
Reset Assertion	N/A	sync, async
Memory File	User input	User Input
Memory File Format	Binary, Hex, Addressed Hex	hex, binary
Enable ECC (not supported for Data Width > 64)	Check box	N/A
Pipeline Stages for Q and ERROR Outputs	0-2	N/A
Bus Ordering Style	Big Endian, Little Endian, None	N/A

PMI

Arithmetic PMI

Table 35: Changes to Arithmetic PMI for CrossLink-NX vs ECP5

PMI	Changes
pmi_add	**.pmi_result_width()** This parameter is not used in CrossLink-NX but retained for compatibility with ECP5 PMI. The user need not specify the result width. The value of "pmi_data_width" is used for both input and output width.
pmi_addsub	.**pmi_result_width()** This parameter is not used in CrossLink-NX but retained for compatibility with ECP5 PMI. The user need not specify the result width. The value of "pmi_data_width" is used for both input and output width.. **Cin** and **Cout** ports are both active-high regardless of the value of Add_Sub. See example in Table 36 below.
pmi_complex_mult	None
pmi_counter	None
pmi_mac	Definition change for pmi_accum_width. It must be wider than (pmi_dataa_width + pmi_datab_width) by 1 to 32 bits.
pmi_mult	None
pmi_multaddsub	None
pmi_multaddsubsum	None
pmi_sub	**.pmi_result_width()** This parameter is not used in CrossLink-NX but retained for compatibility with ECP5 PMI. The user need not specify the result width. The value of "pmi_data_width" is used for both input and output width. **Cin** port is active-high. See example in Table 36 below.

Table 36: Arithmetic PMI Examples

ECP5UM for pmi_addsub:

```
pmi_addsub #
      (.pmi_data_width  (8),
       .pmi_result_width  (8),
       .pmi_sign ("off"),
       .pmi_family ("common"),
       .module_type ( )
       ) u1_addsub8
      (.DataA  (DataA),
       .DataB  (DataB),
       .Cin    ((AddSub) ? Cin : !Cin),
       .Add_Sub(AddSub),
       .Result (Result),
       .Cout   (Cout_int),
       .Overflow()
       );
assign Cout = (AddSub) ? Cout_int : !Cout_int;
```

CrossLink-NX for pmi_addsub:

```
pmi_addsub #(
       .pmi_data_width  (8),
       .pmi_result_width  ( ),
       .pmi_sign ("off"),
       .pmi_family ("common"),
       .module_type ( )
       ) u1_addsub8
      (
       .DataA  (DataA),
       .DataB  (DataB),
       .Cin    (Cin),
       .Add_Sub(AddSub),
       .Result (Result),
       .Cout   (Cout_int),
       .Overflow()
       );
assign Cout = Cout_int;
```

Table 36: Arithmetic PMI Examples (Continued)

ECP5UM for pmi_sub:

```
pmi_sub #
      (.pmi_data_width  (8),
       .pmi_result_width  (8),
       .pmi_sign ("off"),
       .pmi_family ("common"),
       .module_type ( )
       ) u1_sub8
      (.DataA  (DataA),
       .DataB  (DataB),
       .Cin    (!Cin),
       .Result (Result),
       .Cout   (Cout),
       .Overflow()
       );
```

CrossLink-NX for pmi_sub:

```
pmi_sub #(
       .pmi_data_width  (8),
       .pmi_result_width  ( ),
       .pmi_sign ("off"),
       .pmi_family ("common"),
       .module_type ( )
      ) u1_sub8
      (
       .DataA  (DataA),
       .DataB  (DataB),
       .Cin    (Cin),
       .Result (Result),
       .Cout   (Cout),
       .Overflow()
       );
```

Memory PMI

Table 37: Changes to Memory PMI for CrossLink-NX vs ECP5

PMI	Changes
pmi_distributed_dpram	None
pmi_distributed_rom	None
pmi_distributed_shift_reg	None
pmi_distributed_spram	None

Table 37: Changes to Memory PMI for CrossLink-NX vs ECP5 (Continued)

PMI	Changes
pmi_fifo	**.pmi_implementation(), //"LUT", "EBR", "Hard_IP"** New parameter value "Hard_IP" added to Crosslink-NX. When "Hard_IP" value is used, the implementation is faster and uses less LUTs, but may consume additional EBR resources depending on configuration. **.pmi_data_depth()** When "Hard_IP" value is passed through as a parameter for "pmi_implementation", the user is no longer limited by 2^n configurations. **.pmi_full_flag().** This parameter is not a user parameter for Crosslink-NX. The parameter is used in the IP, but the IP would use the value of pmi_data_depth instead. It is suggested to leave the value blank when migrating ECP5 based designs. **.pmi_empty_flag()** This parameter is not a user parameter for Crosslink-NX. The parameter is used in the IP, .pmi_empty_flag() is initialized to "0".
pmi_fifo_dc	**.pmi_implementation(), //"LUT", "EBR", "Hard_IP"** New parameter value "Hard_IP" added to Crosslink-NX. When "Hard_IP" value is used, the implementation is faster and uses less LUTs, but may consume additional EBR resources depending on configuration. No latency is present when using "Hard_IP" and dual-clocks. **.pmi_data_depth_w()** When "Hard_IP" value is passed through as a parameter for "pmi_implementation", the user is no longer limited by 2^n configurations. **.pmi_data_depth_r()** When "Hard_IP" value is passed through as a parameter for "pmi_implementation", the user is no longer limited by 2^n configurations. **.pmi_full_flag().** This parameter is not a user parameter for Crosslink-NX. The parameter is used in the IP, but the IP would use the value of pmi_data_depth instead. It is suggested to leave the value blank when migrating ECP5 based designs. **.pmi_empty_flag()** This parameter is not a user parameter for Crosslink-NX. The parameter is used in the IP, .pmi_empty_flag() is initialized to "0".
pmi_ram_dp	Limitation for Crosslink-NX: cannot be initialized when mixed-width is used. Use pmi_family = "LFD2NX" / "LIFCL" if mixed-width is required, otherwise use pmi_family = "common". (30-bits minimum required for pmi_family = "common").
pmi_ram_dp_be	Limitation: cannot be initialized. User must specify correct device family code in order to be implemented properly.
pmi_ram_dp_true	Limitation for Crosslink-NX: cannot be initialized when mixed-width is used. Use pmi_family = "LFD2NX" / "LIFCL" if mixed-width is required, otherwise use pmi_family = "common". (30-bits minimum required for pmi_family = "common").
pmi_ram_dq	Use pmi_family = "common" if initialization is needed. (30-bits minimum required for pmi_family = "common").

Table 37: Changes to Memory PMI for CrossLink-NX vs ECP5 (Continued)

PMI	Changes
pmi_ram_dq_be	Limitation: cannot be initialized. User must specify correct device family code in order to be implemented properly.
pmi_rom	Use pmi_family = "common". (30-bits minimum required).

Notes:

- All IPs are functionally equivalent to ECP5.
- 30-bit limitation refers to memory size: data_width * addr_depth.

Primitives

The following general guidelines are recommended:

- For primitives that are inferable by the synthesis tool (LUT, I/O, RAM, and so on), let the synthesis tool infer them rather than directly instantiating them.
- If synthesis inference is not sufficient or is not available for a certain primitive, use the Radiant IP Catalog tool rather than directly instantiating the primitive.
- If direct instantiation is required, use the Radiant Source Template tool to help instantiate the primitive. Note that some primitives do not have direct equivalents between Diamond and the Radiant software.

For ease-of-use, you can instantiate the primitives from the Source Template.

For more information on the Radiant primitives, see the Radiant Help under **References > FPGA Libraries Reference Guide**.

Buffers

The Diamond and Radiant software share four basic I/O buffers:

- IB, Input Buffer
- OB, Output Buffer
- OBZ, Output Buffer with Tristate
- BB, Bidirectional Buffer

No changes are needed to migrate these buffers to a Crosslink-NX design. See the following table for coding examples.

Table 38: Buffer Coding Examples

I/O Buffer	Instantiation Examples	Inference Examples
Input Buffer	IB u_IB (.I (reset), // I .O (reset_c) // O);	
Output Buffer	OB u_OB (.I (a_in), // I .O (a_out) // O);	

Table 38: Buffer Coding Examples (Continued)

I/O Buffer	Instantiation Examples	Inference Examples
Output Buffer with Tristate	OBZ u_OBZ (.I (input_i), // I .T (enable), // I, tristate .O (output_o) // O, port pin);	assign output_o = enable ? input_i : 1'bz;
Bidirectional Buffer	BB u_BB (.I (input_i), // I, active-low tristate .T (enable), // I, from fabric .O (out_o), // O, to fabric .B (bidir) // IO, port pin);	assign out_o = enable ? input_i :**3** 1'bz'; assign bidir = out_o;

I/O Registers

The I/O registers inferred in the Radiant software for CrossLink-NX are similar to the ones inferred in Diamond for CrossLink and ECP5. The table below shows the available I/O registers for ECP5, CrossLink, and CrossLink-NX.

Table 39: I/O Registers

Diamond	Radiant Software	Radiant Description
IFS1P3BX	IFD1P3BX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
IFS1P3DX	IFD1P3DX	Positive Edge Triggered Input D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
IFS1P3IX	IFD1P3IX	Positive Edge Triggered Input D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
IFS1P3JX	1FD1P3JX	Positive Edge Triggered Input D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)
OFS1P3BX	OFD1P3BX	Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Preset
OFS1P3DX	OFD1P3DX	Positive Edge Triggered Output D Flip-Flop with Positive Level Enable and Positive Level Asynchronous Clear
OFS1P3IX	OFD1P3IX	Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous Clear and Positive Level Enable (Clear overrides Enable)
OFS1P3JX	OFD1P3JX	Positive Edge Triggered Output D Flip-Flop with Positive Level Synchronous Preset and Positive Level Enable (Preset overrides Enable)

Block RAM (EBR)

In the Radiant software, a block RAM can be configured and generated using IP Catalog, inferred from RTL code, or instantiated using primitives. Instantiating an EBR primitive should be done only if there is a need for a special function.

If a specific size or type of memory function is to be generated, use IP Catalog. For an existing ECP5 design, if the EBR module was generated using Clarity Designer, you need to regenerate the IP using IP Catalog. See "Memory IP" on page 31.

Inferring EBR

EBR blocks inferred by the synthesis tool are totally dependent on the style of the RTL coding. As an example, the following RTL code infers a single-clock pseudo dual port RAM (the PDPSC16K primitive).

```
reg     [7:0] mem [127:0]
reg     [7:0] data_out;

always@(posedge clk) if(we == 1) data_out <= data_out;
 else data_out <= mem[addr];

always @(posedge clk) if (we) mem[addr] <= data_in;
```

To infer a PDP16K primitive, the RTL design must have a read-and-write clock as shown in the example below:

```
wire [8:0]   wr_addr, rd_addr;
wire [7:0]   wr_data;
reg  [7:0]   rd_data;
reg  [511:0] mem [7:0];
always @ (posedge wr_clk)
    if (wr_en) mem[wr_addr] <= wr_data;
always @ (posedge rd_clk)
    if (rd_en) rd_data <= mem[rd_addr];
```

Instantiating EBR Primitives

Instantiation of EBR primitives is only recommended when there is a need for a special function or when different read/write port sizes are required. The table below shows the EBR primitives for CrossLink and ECP5 in Diamond and CrossLink-NX in the Radiant software:

Table 40:

CrossLink	ECP5	CrossLink-NX
DP8KE, PDPW16KD	DP16KD, PDPW16KD	PDP16K, PDP16K_MODE, PDPSC16K, PDPSC512K, SP16K, SP512K, DPSC512K

Instantiate PDP16K, which supports 16Kx1, 8Kx2, 4Kx4, 2Kx9, 1Kx18, or 512x36 memory configurations, using DATA_WIDTH_W and DATA_WDTH_R.

Table 41:

Data Width	Input Data	Output Data	Write Address (MSB to LSB)	Read Address (MSB to LSB)
16Kx1	DI[0]	DO[0]	ADW[13:0]	ADR[13:0]
8Kx2	DI[1:0]	DO[1:0]	ADW[13:1]	ADR[13:1]
4Kx4	DI[3:0]	DO[3:0]	ADW[13:2]	ADR[13:2]
2Kx9	DI[8:0]	DO[8:0]	ADW[13:3]	ADR[13:3]
1Kx18	DI[17:0]	DO[17:0]	ADW[13:4]	ADR[13:4]
512x36	DI[35:0]	DO[35:0]	ADW[13:4]	ADR[13:4]

Below is an instantiation example of PDP16K:

```
PDP16K # (
.DATA_WIDTH_W (8),
.DATA_WIDTH_R (8)
) u_PDP16K(
  .DI   (wr_data_map),              // I, 16-bit data
  .ADW  ({{{2'b0, wr_addr[8:0]}),   // I, 11-bit address
  .ADR  ({2'b0, rd_addr[8:0]}),     // I, 11-bit address
  .CLKW (wr_clk),                   // I
  .CLKR (rd_clk),                   // I
  .CEW  (1'b1),                     // I
  .CER  (1'b1),                     // I
  .CSW  (cs_w),                     // I
  .CSR  (cs_r),                     // I
  .RST  (rst),                      // I
  .DO   (rd_data_map)               // O, 16-bit data
);
```

DSP Functions

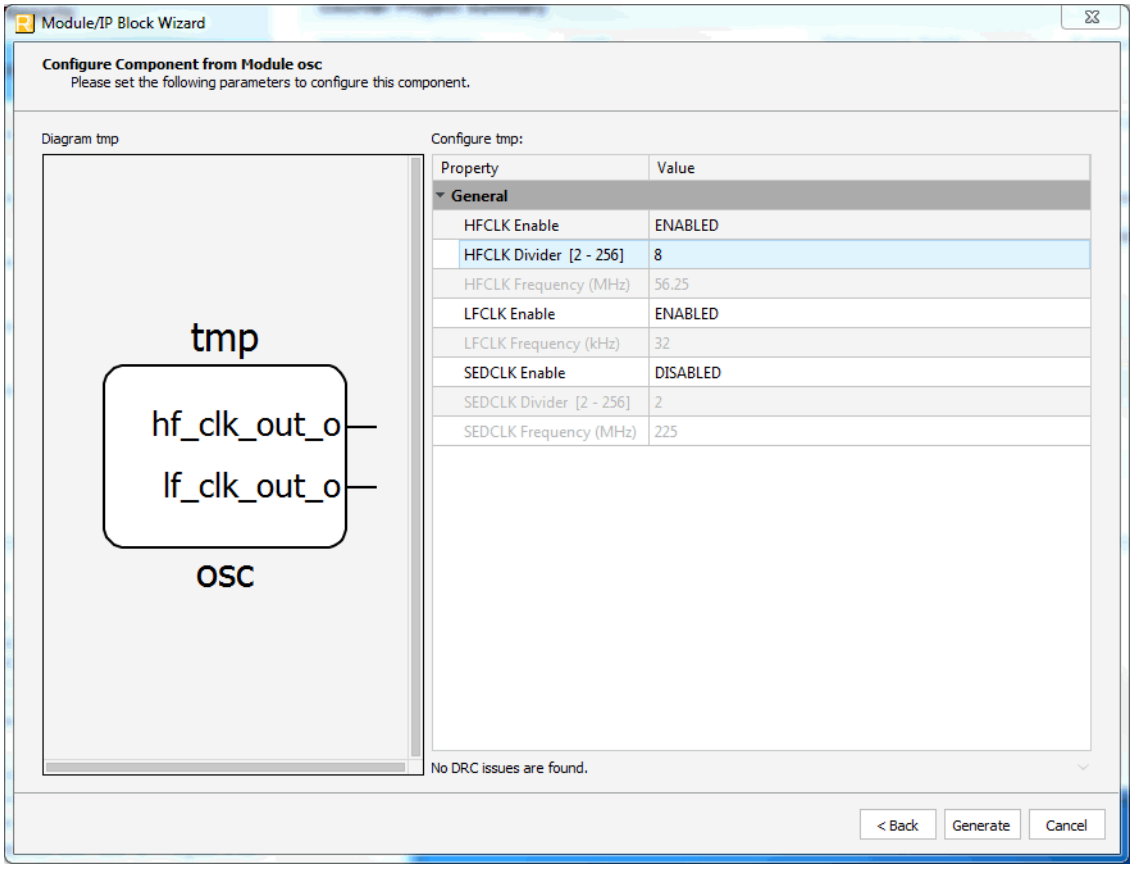
In Diamond, an inferred DSP block remains the same post-synthesis and post-route. But, for CrossLink-NX in the Radiant software, the inferred DSP function changes to hardware primitive models post-synthesis. So it is recommended that you use IP Catalog to generate DSP functions for CrossLink-NX instead of instantiation. See “DSP Arithmetic IP” on page 23.

Oscillator Functions

In Diamond, the oscillator function could be easily instantiated using a primitive. But, in the Radiant software, it is highly recommended that you use IP Catalog to configure and generate the OSC functions.

To use the internal oscillator in CrossLink-NX, use IP Catalog to generate the OSC IP. OSC offers both high-frequency (HFCLK) or low-frequency (LFCLK) clock outputs. Once the IP is configured, generate it and add it into the existing Radiant project. See Figure 1.

Figure 1: OSC Configuration Dialog Box



Registers

In general, registers are inferred by the synthesis tool based on the RTL source code. The Radiant software supports various registers. If you desire a specific type of register function, use Source Template to instantiate it.

Here is an example of instantiating an FD1P3DX (positive-edge-triggered D flip-flop, asynchronous reset, active-high enable) register:

```
FD1P3DX u_ FD1P3DX ( .D (data_in),          // I
                     .SP(en),               // I
                     .CK(clk),              // I
                     .CD(reset),            // I
                     .Q(data_out)           // O
                    );
```

The following table shows a list of registers available for CrossLink and ECP5 in Diamond and for CrossLink-NX in the Radiant software:

Table 42:

Diamond	Radiant Software
FD1P3AX, FD1P3AY, FD1P3BX, FD1P3DX, FD1P3IX, FD1P3JX, FD1S3AX, FD1S3AY, FD1S3BX, FD1S3DX, FD1S3IX, FD1S3JX, FL1P3AY, FL1P3AZ, FL1P3BX, FL1P3DX, FL1P3IY, FL1P3JY, FL1S3AX, FL1S3AY	FD1P3BX, FD1P3DX, FD1P3IX, FD1P3JX, FL1P3AZ

Constraints

One of the more prominent differences between Lattice Diamond software and the Radiant software is the process of constraining a design. In Lattice Diamond, a Logical Preference File (.lpf) was used to constrain a design by tuning all aspects of logical, timing and physical constraints to improve performance.

In the Radiant software, preferences have been replaced by the industry standard Synopsys Design Constraints for ease of use and improved compatibility with third-party vendor tools such as Synopsys Synplify Pro.

Comparing the Constraint Flows

In Lattice Diamond, all preferences are specified post-synthesis. The synthesis flow has its own separate constraining system decoupled from the back end. Any changes in preferences require that the synthesis flow rerun starting from Map Design. See below.

Figure 2: Lattice Diamond Constraints Flow

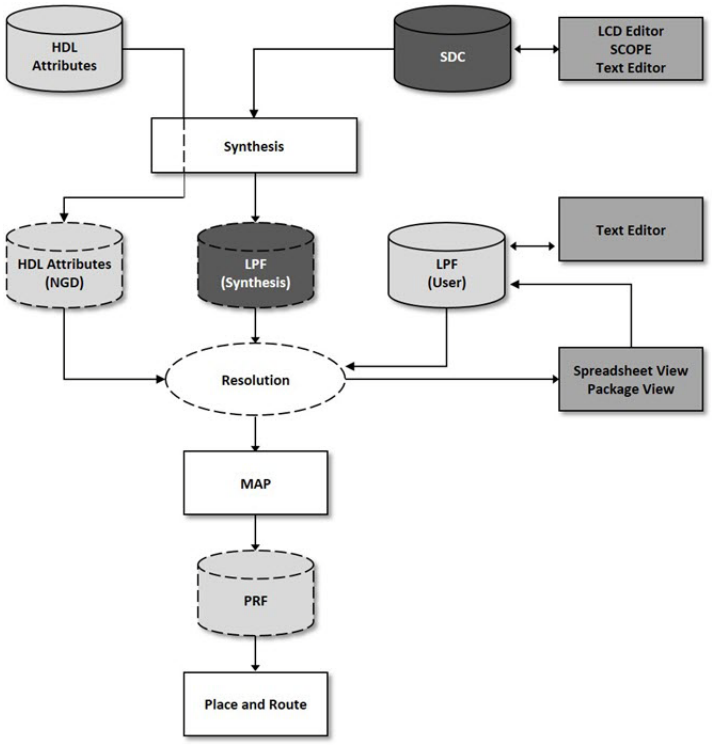
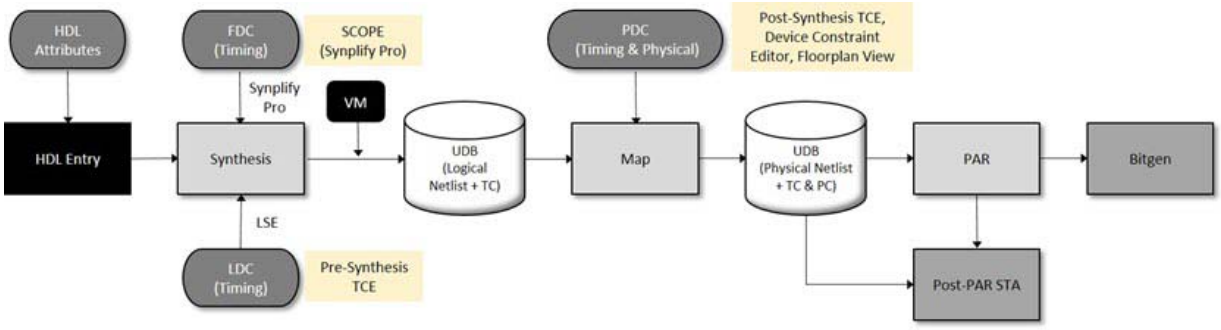


Figure 3 shows how the pre- and post-synthesis constraints are stored in the .ldc file and then kept in the Unified Database (UDB). For other stages in the flow, the PDC holds both post-synthesis timing and physical constraint information. The major difference between this flow and Lattice Diamond is that the Radiant software allows the pre-synthesis constraints to go through the entire flow. You can maintain separate post-synthesis constraints to avoid re-synthesis.

Figure 3: Radiant Constraints Flow



Radiant Constraint Tools

In Lattice Diamond, timing and physical preferences are applied using Spreadsheet View, Package View, Netlist View, and Device View.

In the Radiant software:

- Timing constraints are applied using the Timing Constraint Editors (Pre- and Post-Synthesis).
- Physical constraints are applied using the Device Constraint Editor (DCE).

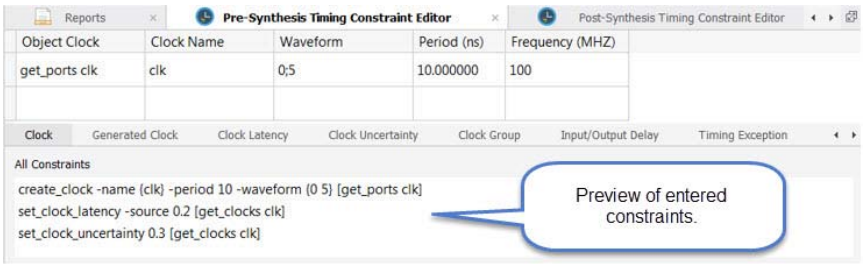
Timing Constraints

As shown in Figure 3, timing constraints are managed in SDC format in an .fdc file for Synopsys Synplify Pro synthesis or in an .ldc file for Lattice LSE synthesis.

The new Radiant software tools for pre- and post-synthesis timing constraints are:

- Pre-Synthesis Timing Constraint Editor (Figure 4)—Reads the HDL designs and helps you to create timing constraints based on HDL signal, port, and object names. The constraints are saved in an .ldc file.

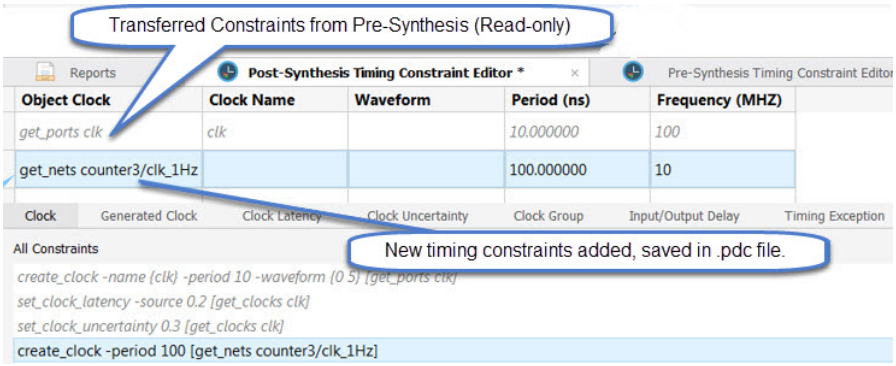
Figure 4: Pre-Synthesis Timing Constraint Editor



- Post-Synthesis Timing Constraint Editor (Figure 5)—Reads the post-synthesis netlists and helps you to create timing constraints based on post-synthesis netlist signal, port, and object names. The constraints are

saved in a .pdc file. This flow allows further tuning of timing constraints for the Place & Route process.

Figure 5: Post-Synthesis Timing Constraint Editor



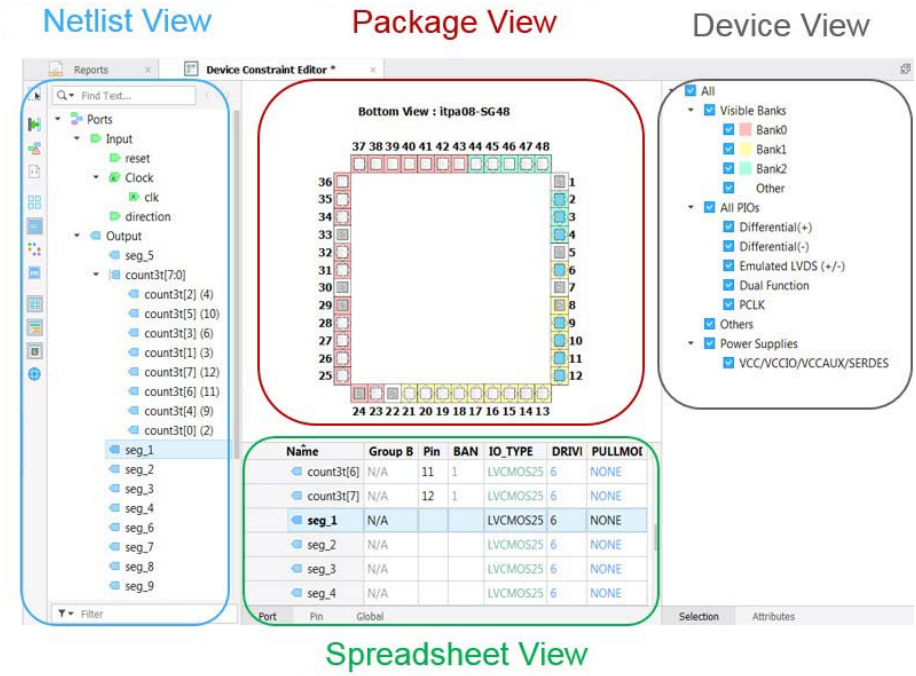
The updated post-synthesis timing constraints override pre-synthesis constraints. This happens only when the same constraint is applied in post-synthesis.

In general, physical constraints are entered in the .pdc file. Alternatively, through a text editor, the physical constraints may also be entered in the .ldc file. Synthesis does not consume them and transfer them to a .pdc file for back-end consumption.

Physical Constraints

The Radiant Device Constraint Editor tool now combines the Netlist, Package, Device, and Spreadsheet views into one GUI for the primary purpose of entering physical (.pdc) constraints. This makes it easier to manage multiple tools and perform such actions as cross probing between multiple views. See Figure 6.

Figure 6: Device Constraint Editor



Each of the views are used to apply constraints such as prohibiting pins and assigning IO_TYPEs.

Note that the Floorplan View is used for creating GROUPs and REGIONs. For more information on Device Constraint Editor, see the Radiant software online Help under **User Guides > Applying Design Constraints > Using Radiant Software Tools > Device Constraint Editor.**

Preferences to Constraints

Table 43 shows the most commonly used Lattice Diamond physical preference keywords and the equivalent Radiant SDC commands to create a physical constraint.

Table 43: Lattice Diamond Preference Keywords Compared to the Radiant SDC Commands

Lattice Diamond Physical Preference	Radiant Software Physical Constraint	Description
Global, Net, and Clock Attributes	ldc_set_attribute	Sets global attributes or specific attributes to the selected object.
UGROUP	ldc_create_group	Defines a user group.
IOBUF	ldc_set_port	Sets constraint attributes to the selected port.
LOCATE	ldc_set_location	Places an object on a site or a user group into a region.

Table 43: Lattice Diamond Preference Keywords Compared to the Radiant SDC Commands (Continued)

Lattice Diamond Physical Preference	Radiant Software Physical Constraint	Description
LOCATE VREF	ldc_create_vref	Defines a voltage reference site.
PROHIBIT	ldc_prohibit	Prohibits use of a site.
REGION	ldc_create_region	Defines a rectangular region.
SYSCONFIG	ldc_set_sysconfig	Sets SysConfig attributes.
VCC_NOMINAL VCC_DERATE VOLTAGE	ldc_set_vcc	Sets a voltage to a bank or derates the core voltage.

All physical constraints and post-synthesis timing constraints are stored in the .pdc file.

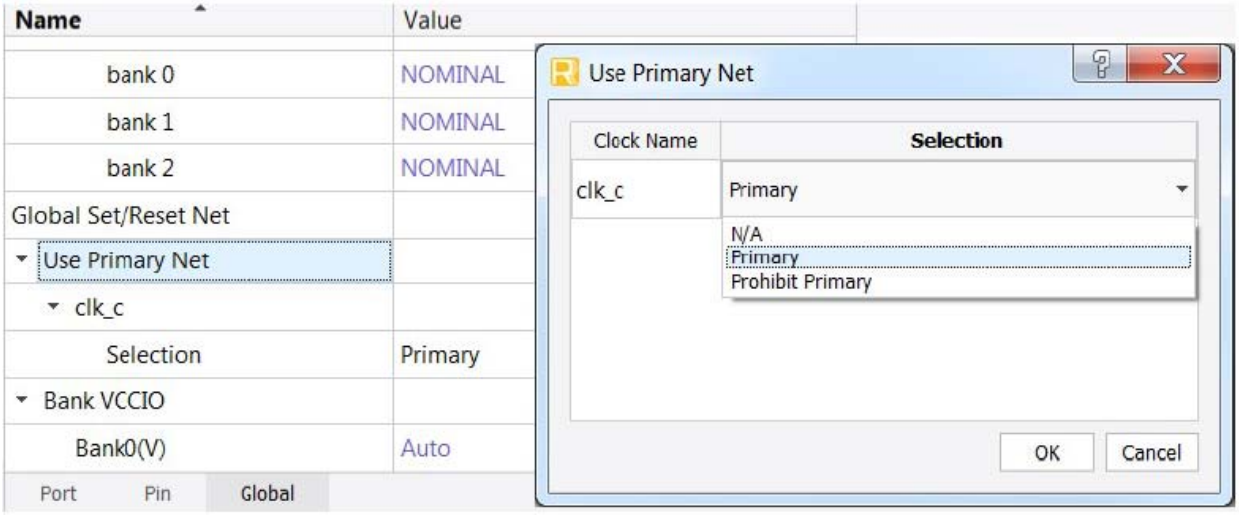
Primary Clock Net Access

In Diamond, primary clocks are specified in a preference file (.lpf) but in the Radiant software, the primary clocks are specified in an .ldc file. In the Radiant software, you can use Device Constraint Editor to set access to the primary clock spine.

To set access to the primary clock spine:

1. Choose **Tools > Device Constraint Editor**.
2. Click the **Global** tab.
3. Double-click on **Use Primary Net**.

   The Use Primary Net dialog box opens.



4. Double-click on the desired clock net.
5. Choose **Primary** or **Prohibit Primary** from the pull-down menu.
6. Click **OK**.

Once the change is saved, the attribute is recorded as shown below:

```
#Set to use Primary clock spine
ldc_set_attribute {USE_PRIMARY=TRUE} [get_nets clk_c]

#Prohibit primary clock spine
ldc_set_attribute {USE_PRIMARY=FALSE} [get_nets clk_c]
```

Timing Preferences to Constraints

Table 44 shows the most commonly used Lattice Diamond timing preference keywords and the equivalent Radiant SDC timing constraints.

Table 44: Diamond Timing Preference Keywords Compared to Radiant SDC Timing Constraints

Lattice Diamond Timing Preference	Radiant Software Timing Constraint	Description
BLOCK INTERCLOCK DOMAIN	set_clock_groups	Defines different types of clocking schemes.
BLOCK CLKNET BLOCK PATH CLKSKEWDISABLE	set_false_path	Defines false path cycles.
CLKSKEWDIFF	set_clock_latency	Defines arrival and departure times.
CLOCK_TO_OUT	set_output_delay	Defines output delay relative to a clock.
FREQUENCY/PERIOD	create_clock	Defines the design clocks.
FREQUENCY/PERIOD	create_generated_clock	Defines generated clocks.
INPUT_SETUP	set_input_delay	Defines input delay relative to a clock.
MAX_DELAY	set_max_delay	Defines maximum delay for timing paths.
MAX_DELAY MIN	set_min_delay	Define minimum delay for timing paths.
MULTICYCLE	set_multicycle_path	Defines multicycle clock cycles.
SYSTEM_JITTER CLOCK_JITTER (option)	set_clock_uncertainty	Defines uncertainty delays.

Note

When an INPUT_SETUP or CLOCK_TO_OUT is set in Lattice Diamond without first specifying a clock, Lattice Diamond automatically creates a virtual clock constraint and honors the delay. In the Radiant software, you should first define a clock (create_clock), whether real or virtual, before using the set_input_delay or set_output_delay constraint.

The Radiant software constraints listed in Table 45 require Tcl commands to access object names such as cell, pin, net, port, or clock in a design.

Table 45: Tcl Commands

Object Access Types	Description
all_clocks	Access all clocks in a design.
all_inputs	Access all inputs in a design.
all_outputs	Access all outputs in a design.
get_cells	Access cells in a design.
get_clocks	Access clocks in a design.
get_nets	Access nets in a design.
get_pins	Access pins in a design.
get_ports	Access ports in a design.

Table 8 lists examples commonly used in Lattice Diamond preferences and the equivalent Radiant software constraints in the SDC format.

Table 46: Examples of Timing Preferences in SDC Format

Lattice Diamond Preference	Radiant Software Constraint
BLOCK PATH FROM PORT “abc” TO CELL “reg1/”;	set_false_path -from [get_ports abc] -to [get_cells reg1]
CLKSKEWDIFF CLKPORT “clk1” CLKPORT “clk2” 2 NS;	set_clock _latency 2 -source [get_clocks clk1]
CLKSKEWDISABLE CLKNET “clk1” CLKNET “clk2”;	set_false_path -from [get_clocks clk1] -to [get_clocks clk2]
CLOCK_TO_OUT PORT “out1” 8 ns CLKPORT=”clk2”;	set_output_delay (x-8) -clock [get_clocks clk2] [get_ports out1] [1, 2]
FREQUENCY (PERIOD) NET “clk1” 100Mhz;	create_clock -period 10 -name clk1 [get_nets clk1]
INPUT_SETUP PORT “in_a” 4 ns CLKNET ”clk1”;	set_input_delay (x-4) -clock [get_clocks clk1] [get_ports_in_a] [1, 2]
MAXDELAY FROM CELL “reg1” TO CELL “reg2” 5 NS	set_max_delay 5 -from [get_cells reg1] -to [get_cells reg2]
MULTICYCLE FROM CLKNET “clk1” TO CLKNET “clk2” 2 X;	set_multicycle_path 2 -from [get_clocks_clk1] -to [get_clocks clk2] [2]
SYSTEM_JITTER 1.0 NS	set_clock_uncertainty 1 [get_clocks *]

1. x = clock period

2. Set create_clock first.

For more information on the details of SDC constraints, see the Radiant Help under **Reference Guides > Constraints Reference Guide > Lattice Synthesis Engine Constraints > Synopsys Design Constraints.**

Attributes Compared

Synthesis attributes are mostly the same in both Lattice Diamond and the Radiant software. The Radiant software does not use some attributes because they are only for Diamond's preference method. Also, some attributes are for devices not supported in the Radiant software. Any attributes not in Table 47 (below) that were in Lattice Diamond are obsolete and are not used in the Radiant software.

Table 47: Diamond versus Radiant Attributes

Lattice Diamond Software	Radiant Software
BBOX	BBOX
CLAMP	CLAMP
DIFFRESISTOR	DIFFRESISTOR
DRIVE	DRIVE
GLITCHFILTER	GLITCHFILTER
GSR	GSR
HGROUP	Replaced by GRP
HYSTERSIS	HYSTERSIS
INIT	INIT
IO_TYPE	IO_TYPE
LOC	LOC
NOCLIP	NOCLIP
NOMERGE(SAVE)	NOMERGE
OPENDRAIN	OPENDRAIN
PULLMODE	PULLMODE
RBBOX	RBBOX
REGION	REGION
SLEWRATE	SLEWRATE
TERMINATION	TERMINATION
UGROUP	Replaced by GRP

Table 47: Diamond versus Radiant Attributes (Continued)

Lattice Diamond Software	Radiant Software
USERCODE	USERCODE
VREF	VREF

Comparing Diamond and the Radiant Software

A primary feature of the Lattice Radiant™ software is to enable Lattice Diamond users to import their field programmable gate array (FPGA) designs easily and seamlessly. Lattice Diamond users will quickly recognize a familiar design environment, graphical user interface (GUI) layout, and related tools such as Power Calculator and Reveal. However, the Radiant software includes many significant feature enhancements from Lattice Diamond, including:

- Upgraded database to provide fast and efficient software.
- IEEE 1735 standard support for encryption flow.
- Constraints utilizing industry standard Synopsys Design Constraints (SDC) format.
- Efficient and easy-to-use GUI.
- Unified timing analysis engine for faster design timing closure.

The objective of this guide is to help Lattice Diamond users quickly grasp the concepts of the new features in the Radiant software.

The following table lists major functions in a typical FPGA design flow, and compares Lattice Diamond to the Radiant software.

Table 48: Comparison of Lattice Diamond to the Radiant Software

	Lattice Diamond Design Functions	Radiant Software Design Functions
Design Entry	Supports hardware design languages (HDL) such as Verilog and VHDL.	Supports Verilog and VHDL. In addition, the Radiant software provides the Source Template tool, which is an enhanced version of Diamond's Template Editor. Source Template helps you implement Verilog or VHDL language constructs, device primitives, and PMI templates more efficiently. See "Design Entry" on page 62.
	Supports primitives instantiation.	Supports primitives instantiation. See "Using Radiant Primitives" on page 63.
	Supports soft IP and modules with IP Express and Clarity Designer tools.	Supports soft IP and modules with IP Catalog. See "Using Modules and Soft IP in the Radiant Software" on page 63.
	Supports Parameterized Module Instantiation (PMI).	Supports PMI. See "Using Parameterized Module Instantiation" on page 65.
	Supports EDIF flow.	Supports VM flow.
Constraints	All timing and physical constraints defined in preferences.	All timing and physical constraints defined in standard SDC. See "Radiant Constraint Tools" on page 52.
	Supports HDL attributes.	Supports HDL attributes. See "Design Implementation" on page 66.
	Supports preferences flow.	Supports constraints flow. See "Design Implementation" on page 66.
Implementation	Supports standard FPGA design flow including synthesis, map, place-and-route (PAR), and bitgen.	Supports standard FPGA design flow including synthesis, map, PAR, and bitgen. See "Lattice Diamond and the Radiant Process Flow" on page 70.
	Process flow supported.	Process flow is implemented in Process Toolbar. See "Lattice Diamond and the Radiant Process Flow" on page 70.
	Supports strategies, which are options related to implementation tools.	Supports strategies. See "Design Analysis and Debug" on page 71.
	Project file extension is .ldf.	Project file extension is .rdf. See "Starting the Import Diamond Project Wizard" on page 67.

Table 48: Comparison of Lattice Diamond to the Radiant Software (Continued)

	Lattice Diamond Design Functions	Radiant Software Design Functions
Analysis and Debug	Includes Reveal Inserter and Reveal Analyzer debug and analysis tools.	Includes Reveal Inserter and Reveal Analyzer debug and analysis tools. Reveal Inserter also has a new Reveal Controller module that allows you to monitor and control status registers in hard IP such as I2CFIFO, PLL, PCIe, CDR, and DPHY. See "Reveal" on page 71.
	Includes Timing Analysis View.	Includes new Timing Analysis View with new constraints flow including Setup/Hold Endpoint Analysis. See "Design Analysis and Debug" on page 71.
	Includes Power Calculator.	Includes Power Calculator. See "Power Calculator" on page 71.
	Includes Simulation Wizard.	Includes Simulation Wizard with the support of Post-Synthesis Simulation. See "Simulation Wizard" on page 71.
	Models of protected primitives are provided in the form of a pre-compiled library (black box). Only certain simulators (Active-HDL and ModelSim) could be supported.	Models are provided in Verilog source form with IEEE P1735 encryption. This allows all third-party tools that support the same standard to be supported for Radiant 2.0. See Lattice Radiant Software 2.0 Release Notes for a list of supported third-party tools.
Tools	Includes a full suite of Lattice Diamond tools.	Includes a full suite of Radiant tools. Some tools have been combined for more efficiency. See "Radiant Software Tools" on page 78.

Design Entry

Hardware description languages (HDLs) such as Verilog and VHDL are fully supported in both Lattice Diamond and the Radiant software. This section describes various design HDL methodologies used in both Lattice Diamond and the Radiant software, and highlights similarities and differences.

Using HDL in the Radiant Software

You can re-use your Verilog and VHDL source files when migrating designs from Lattice Diamond to the Radiant software.

Using Radiant Primitives

If your design contains Lattice Diamond primitives, you may need to replace them with new Radiant primitives. For more information, refer to the *FPGA Libraries Reference Guide* in the Radiant online Help.

Using Modules and Soft IP in the Radiant Software

The IPexpress/Clarity tools in Lattice Diamond support two types of functional blocks: modules and intellectual property (IP). In the Radiant software, modules and IP can be created as part of a specific project or as a library for multiple projects.

- Modules are basic, configurable blocks that provide a variety of functions including I/O, arithmetic, memory, and more.
- Soft IP are more complex, configurable blocks that must first be downloaded and installed as a separate step before they can be accessed.

The major difference between Lattice Diamond and the Radiant software is the soft IP flow as shown in the following figures.

Figure 7: Lattice Diamond IPexpress/Clarity Flow

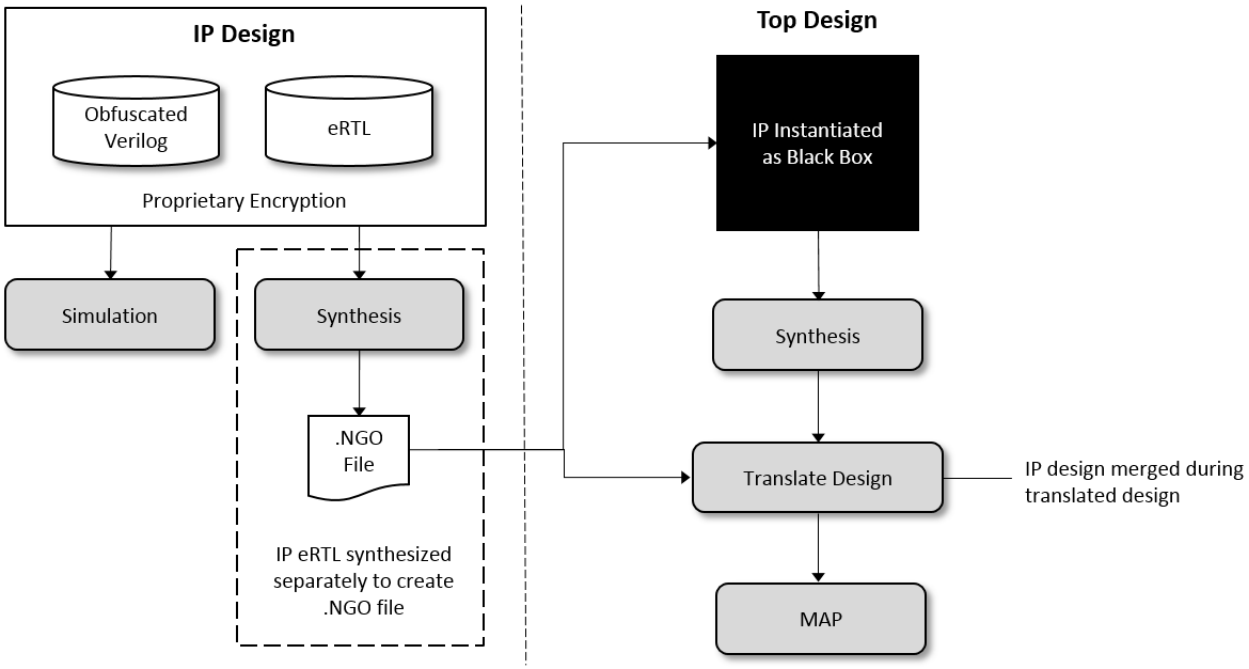
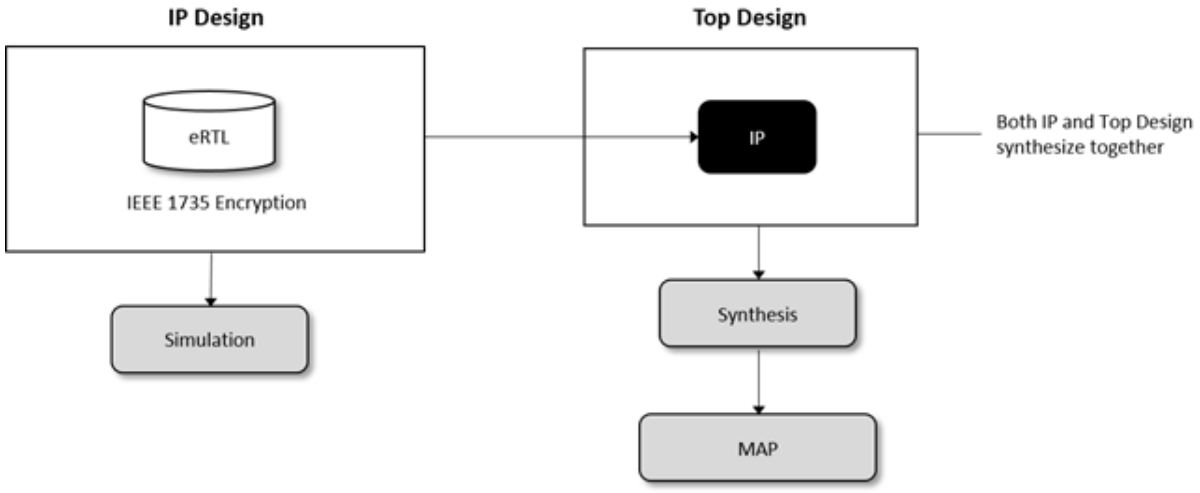


Figure 8: Radiant Soft IP Flow



The following table lists differences between the Lattice Diamond and the Radiant soft IP and module flows.

Table 49: Diamond versus Radiant Soft IP and Module Flows

Lattice Diamond	Radiant Software
A simulation testbench file is included when IP is generated.	A simulation testbench file is included when IP is generated. In module generation, a simulation testbench file is optional.
Uses Blowfish encryption.	Uses IEEE 1735 encryption.
▶ Encrypted eRTL file used for synthesis. ▶ Obfuscated Verilog used for simulation.	Encrypted modules/IP can be synthesized and simulated.
Many IP output files are generated.	Fewer IP output files are generated. Flow simplified in the Radiant software.

Radiant IP Catalog Output Files for Modules IP Catalog generates the output files shown in the following table for modules under the specified Project Path. The *<file_name>* comes from the file name specified in the Configuration tab.

Table 50: Radiant Module IP Output Files

File Type	Definition	Function
<file_name>.ipx	IP manifest file	Used to list all files generated by IP generation engine.
<file_name>.v	Verilog module file	Verilog netlist generated by IP Catalog to match the configuration of the module.

Table 50: Radiant Module IP Output Files (Continued)

File Type	Definition	Function
<*file_name*>_bb.v	Verilog IP black box file	Provides the Verilog synthesis black box for the IP core and defines the port list.
<*file_name*>.cfg	IP parameter configuration file	Used for re-creating or modifying the core in IP Catalog.
<*file_name*>.xml	IP metadata file	Describes the legal usage and interface of the soft IP.
<*file_name*>_tmpl.v	Verilog template file	A template for instantiating the generated module. This file can be copied into a Verilog file.
<*file_name*>_tmpl.vhd	VHDL module template file	A template for instantiating the generated module. This file can be copied into a VHDL file.

Using Parameterized Module Instantiation

Parameterized Module Instantiation (PMI) is an alternate way to use some of the modules in Lattice Diamond and the Radiant software. PMI allows you to directly instantiate a module into your HDL and customize it by setting parameters in the HDL.

The following table lists the differences in Lattice Diamond PMI parameters versus the Radiant PMI parameters.

Table 51: Lattice Diamond PMI Compared to the Radiant PMI

Category	PMI	Lattice Diamond	Radiant Software
Architecture	pmi_pll	Yes	No
	pmi_pll_fp	Yes	No
Arithmetic	pmi_add	Yes	Yes
	pmi_addsub	Yes	Yes
	pmi_complex_mult	Yes	Yes
	pmi_constant_mult	Yes	No
	pmi_counter	Yes	Yes
	pmi_mac	Yes	Yes
	pmi_mult	Yes	Yes
	pmi_multaddsub	Yes	Yes
	pmi_multaddsubsum	Yes	Yes
	pmi_sub	Yes	Yes

Table 51: Lattice Diamond PMI Compared to the Radiant PMI (Continued)

Category	PMI	Lattice Diamond	Radiant Software
DSP Arithmetic	pmi_dsp	No	Yes, iCE40 UltraPlus only
	pmi_dsp_casmultaddsub	Yes	No
	pmi_dsp_mac	Yes	No
	pmi_dsp_mult	Yes	No
	pmi_dsp_multaddsub	Yes	No
	pmi_dsp_multaddsubsum	Yes	No
	pmi_dsp_preadd_slice	Yes	No
FIFO	pmi_fifo	Yes	Yes
	pmi_fifo_dc	Yes	Yes
	pmi_fifo_dp_be	No	Yes
	pmi_fifo_dp_true_be	No	Yes
	pmi_fifo_dq_be	No	Yes
Distributed RAM	pmi_distributed_dpram	Yes	Yes
	pmi_distributed_rom	Yes	Yes
	pmi_distributed_shift_reg	Yes	No
	pmi_distributed_spram	Yes	Yes
EBR	pmi_ram_dp	Yes	Yes
	pmi_ram_dp_be	Yes	Yes
	pmi_ram_dp_true	Yes	No
	pmi_ram_dp_true_be	Yes	No
	pmi_ram_dq	Yes	Yes
	pmi_ram_dq_be	Yes	Yes
	pmi_rom	Yes	Yes

For more information on the Radiant soft IP and modules, refer to the following documents:

- Arithmetic Modules User Guide
- Memory Modules User Guide

Design Implementation

This section describes how to import a Lattice Diamond project into the Radiant software. This section also describes similarities and differences between Lattice Diamond and the Radiant software.

- Lattice Diamond project files are not compatible with the Radiant project files. Lattice Diamond project files use .ldf extension. The Radiant project files use .rdf extension.
- The Radiant software provides a wizard that allows you to import a Lattice Diamond project into the Radiant software.

Importing a Lattice Diamond Project into the Radiant Software

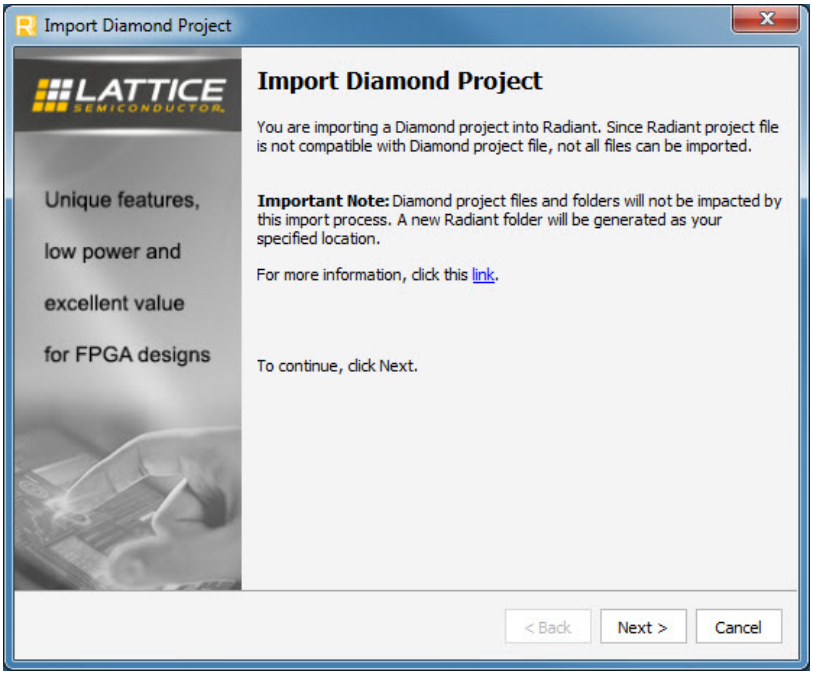
The Radiant software allows you to import Lattice Diamond projects using the Import Diamond Project wizard. Many features and settings from Lattice Diamond can be directly imported into the Radiant software, allowing you to avoid creating a new project from scratch.

Starting the Import Diamond Project Wizard

Design projects created in Lattice Diamond can be imported into the Radiant software using the Import Diamond Project wizard. Imported Lattice Diamond projects are targeted to devices supported in the Radiant software. Lattice Diamond design preferences will be converted into the Radiant software design constraints.

To import a Lattice Diamond Project into Radiant software, open the Radiant software. Then, choose **File > Open > Import Diamond Project**.

Figure 9: Import Diamond Project Wizard



For more information on using the Import Lattice Diamond Project wizard, in the Radiant Help, see **User Guides > Managing Projects > Importing Lattice Diamond Projects**.

Importing Lattice Diamond Strategies into the Radiant Software

A strategy provides a unified view of all the options related to implementation tools such as synthesis, map, and place and route. Strategy options are listed in the Strategy dialog box in both Lattice Diamond and the Radiant software.

Lattice Diamond strategies are imported into the Radiant software if the strategies are unchanged. If the Radiant software does not support Lattice Diamond strategies, those Lattice Diamond strategies will be ignored. You should check all strategies after they have been imported from Lattice Diamond into the Radiant software.

Compatible Settings and Files

The following Lattice Diamond settings and source can be imported into the Radiant software:

HDL Source Files and Properties HDL source files such Verilog (.v) and VHDL (.vhd) can be imported into the Radiant software.

Synplify Pro SDC Files Synplify Pro Timing Constraint Files (SDC) can be imported into the Radiant software.

Lattice Synthesis Engine Timing Constraint File LSE timing constraints can be imported into the Radiant software. Some records with buses do not use standard Tcl syntax and are discarded when loaded into the Radiant database.

Implementations All implementations and their settings, such as top-level unit and synthesis tool selection, are imported into the Radiant software.

Strategies All supported strategies are imported into the Radiant software.

Reveal Inserter Settings Reveal Inserter debug files (.rvl) are imported from Lattice Diamond into the Radiant software. However, due to differences between devices, not all options or features may be available in the Radiant Reveal Inserter.

Simulation Wizard All Simulation Wizard scripts are imported into the Radiant software.

Incompatible Settings and Files

Some settings and files from Lattice Diamond are not compatible with the Radiant software. The following Lattice Diamond settings and files cannot be imported:

Target Device Currently no devices supported by Lattice Diamond are supported in the Radiant software. Therefore, it will be necessary to select a Radiant software-supported device.

Lattice Diamond Preference (.lpf) Files Lattice Diamond preference files (.lpf) are not imported into the Radiant software. The Radiant software does not support Lattice Diamond preferences. All Lattice Diamond preferences must be manually converted into the Radiant software constraints.

Soft IP and Module (.ipx) Files Soft IP and Module package instance files (.ipx and .sbx) are not imported because the IP flow in the Radiant software differs from Lattice Diamond. Refer to "Using Modules and Soft IP in the Radiant Software" on page 63 for a description of the differences.

Reveal Analyzer File Because of differences between devices, Lattice Diamond Reveal Analyzer (.rva) files imported into the Radiant software may not be useful. New waveforms generated with the Radiant Reveal Analyzer will overwrite .rva files imported from Lattice Diamond.

Power Calculator File Because of differences between devices, Lattice Diamond Power Calculator (.pcf) files cannot be imported into the Radiant software.

Programmer Project File Because of differences between devices, Programmer Project Files (.xcf) files cannot be imported into the Radiant software.

Unsupported Design Source in Radiant Software

The following design source files are not supported in the Radiant software and cannot be imported:

EDIF Design Entry Files Electronic Design Interchange Format (EDIF) is not supported in the Radiant software. Therefore, such files as .edf and .edn cannot be imported into the Radiant software.

Schematic Files The Radiant software does not support schematic entry. Therefore, schematic files cannot be imported into the Radiant software.

Clarity Design Files The Radiant software does not support Clarity Designer (.sbx) files. Therefore, .sbx files cannot be imported into the Radiant software.

Platform Designer Files The Radiant software does not support Platform Manager or Platform Manager 2 devices. Therefore, Platform Designer (.ptm) files cannot be imported into the Radiant software.

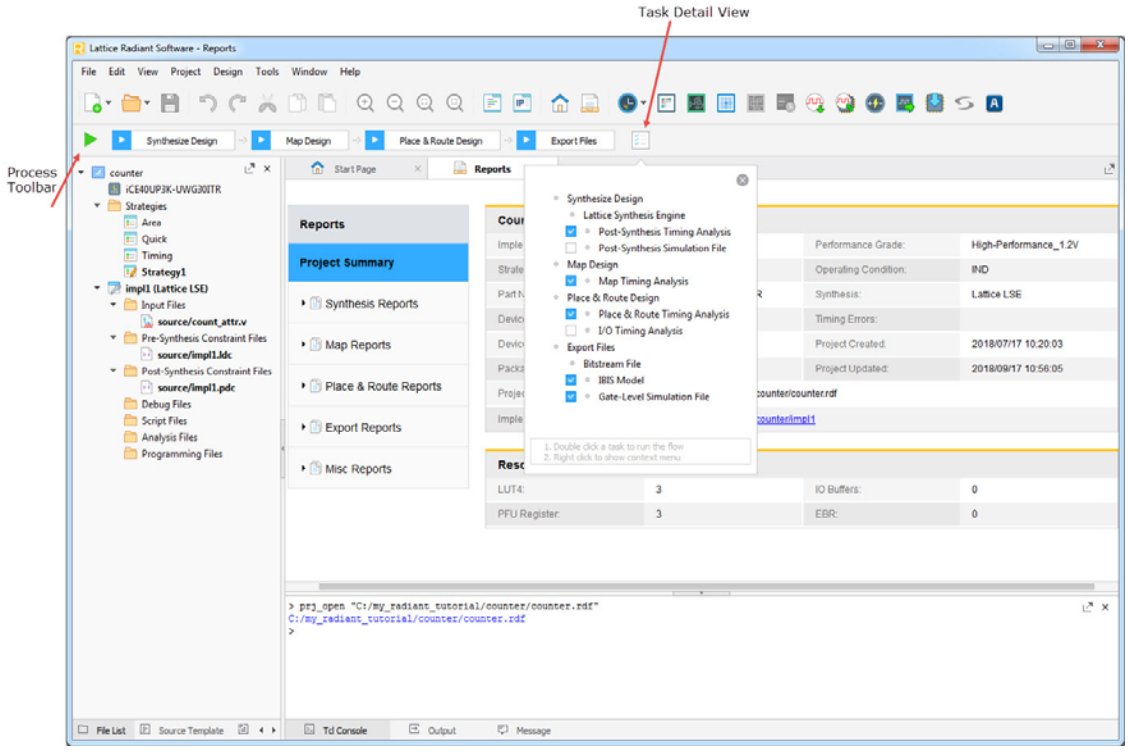
Lattice Diamond and the Radiant Process Flow

The Radiant graphical user interface has changed from the Lattice Diamond. The Radiant main window is shown in Figure 10.

Processes are controlled using the Process Toolbar. All process tasks can be viewed and selected using the Task Detail View.

- The Radiant software also supports Lattice Diamond project features such as Implementations and Strategies.
- A major difference between Lattice Diamond and the Radiant software is the constraint language and flow, as discussed in "Design Implementation" on page 66.
- Report View in the Radiant software is similar to Report View in Lattice Diamond, but the Radiant software uses different report formats.

Figure 10: Radiant Software Main Window



Design Analysis and Debug

With the exception of the Simulation Wizard and Timing Analysis tools, the debug and analysis tools within the Radiant software have remained unchanged. For more information on Timing Analysis, refer to "Timing Analysis" on page 74.

Simulation Wizard

The Simulation Wizard concept remains the same. You are guided through the simulation set-up process as before.

The Radiant software now supports post-synthesis simulation. You are able to perform a simulation of a post-synthesis netlist (<*file_name*>_syn.vo) file. In order to generate this file, enable the Post-Synthesis Simulation File option in Task Detail View and then synthesize the design.

Support for Post-Place and Route back-annotated simulation is consistent with Lattice Diamond. The Post-Map back-annotated simulation is not supported by the Radiant software.

Lattice Diamond	Radiant Software
Process stage supported by Simulation Wizard: ▶ RTL Simulation ▶ Post-Map Gate-Level Simulation ▶ Post-Route Gate-Level + Timing Simulation	Process stage supported by Simulation Wizard: ▶ RTL Simulation ▶ Post-Synthesis Simulation ▶ Post-Route Gate-Level Simulation ▶ Post-Route Gate-Level + Timing Simulation

First 22 ns of simulation is invalid.

Power Calculator

Power Calculator is unchanged from Lattice Diamond.

Reveal

The Reveal Inserter and Reveal Analyzer are the same as in Diamond but with the addition of a Reveal Controller module. This new type of module enables:

- Access to the control and status registers of the hard IPs
- Virtual switches and LEDs to control and monitor the design
- Read and write access to a bank of user registers and memory

Timing Analysis

The following table lists differences between Lattice Diamond and the Radiant software.

Lattice Diamond	Radiant Software
Timing is modeled as register-to-register within the FPGA boundary. Constraints are modeled in four methods: ▶ Input to output paths (MAXDELAY) ▶ Input to register paths (INPUT_SETUP) ▶ Register to register paths (FREQUENCY/PERIOD) ▶ Register to output paths (CLOCK_TO_OUT)	Timing is modeled register-to-register from outside of the FPGA boundary. ▶ Register to register paths (create_clock, set_input/output_delay, set_clock_latency)
HTML links to timing paths.	▶ HTML-based table of contents for easy navigation. ▶ Cross-probing from reports. You can click on a hyperlink icon to cross probe into that tool. For example, the Post-Synthesis & Map timing report links to Netlist Analyzer. In the PAR timing report, you can cross-probe to Netlist Analyzer, Floorplan View, and Physical View.
Critical Endpoints: Critical paths listed for each single path analyzed. (Need to analyze every path to determine critical endpoints.)	Lists top critical endpoints irrespective of the constraint or path.
Embedded within some of the timing reports.	Unconstrained clocks and paths clearly listed.

Static Timing Analysis Concepts in Lattice Diamond and the Radiant Software

Lattice Diamond modeled timing paths from within the FPGA boundary. This explains why an assortment of timing preferences were available to model timing paths that began and ended within the FPGA. Preference constraints such as CLOCK_TO_OUT, INPUT_SETUP and MAXDELAY were used to constrain these paths with the assumptions that any timing information for arriving exterior logic is known.

Figure 11: Lattice Diamond Constraint Timing Modeling

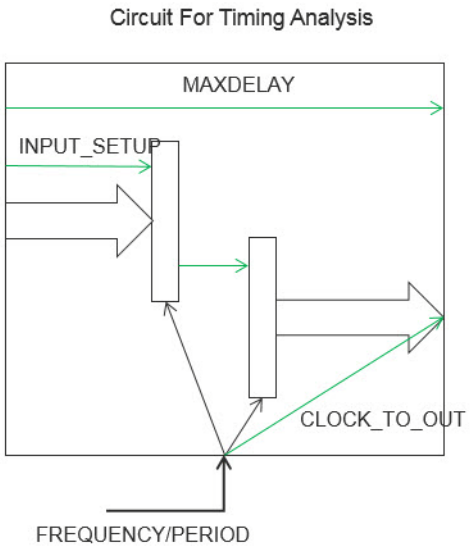
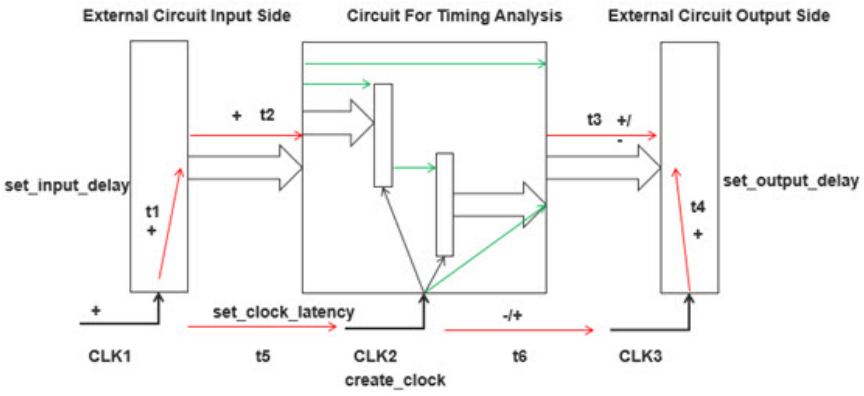


Figure 11 shows an overview of how a circuit is modeled for timing within the Radiant software. As mentioned earlier, the key difference with the Radiant software is that the path is modeled from register-to-register across the entire path which can begin and end outside of the FPGA boundary. As a result, standard SDC constraints are used to model the input and output delays (that is, set_input/output_delay) from external to the FPGA boundary, and appropriate latency (ldc_set_latency) can be specified from external to internal of the chip boundary.

Critical endpoints list the paths for clocking constraints that have no departure or arrival element.

Unconstrained endpoints show paths that have no constraints defined even though the path is clocked, indicating that there is no departure, arrival primitive or port.

Figure 12: Sample Circuit Analysis



For more information on Device Constraint Editor, refer to the Radiant Help topic **User Guides > Analyzing Static Timing > Using Timing Analysis View > Timing Analysis View Feature > Timing Analysis View Main Window Tabs**.

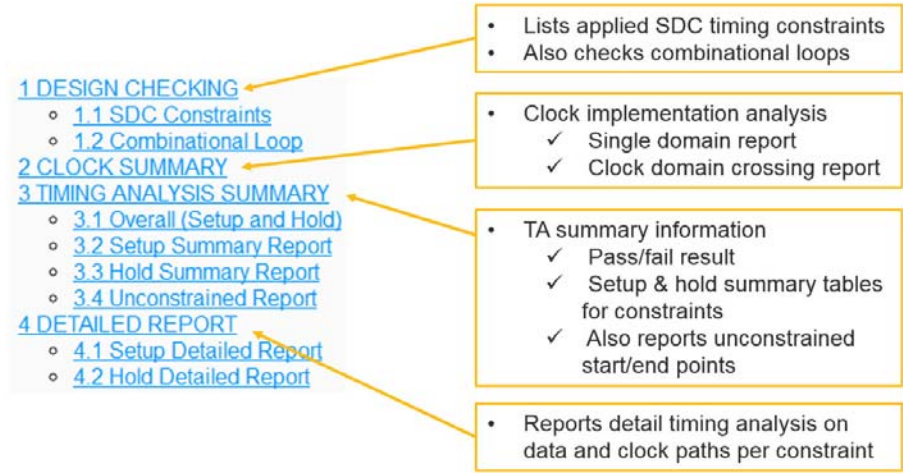
Timing Analysis

The Radiant software makes it easier to analyze and navigate design constraints. In addition to Preference Reports being replaced by .sdc constraints, Timing Reports are now organized in a table of contents with click-able .html links that take you to a specific section for analysis.

Once static timing analysis is performed, .twr and .html report files are created. The report structure of the .sdc constraints is also different than in Lattice Diamond.

The Radiant software Timing Report is grouped into multiple hyper-linked categories, as shown below.

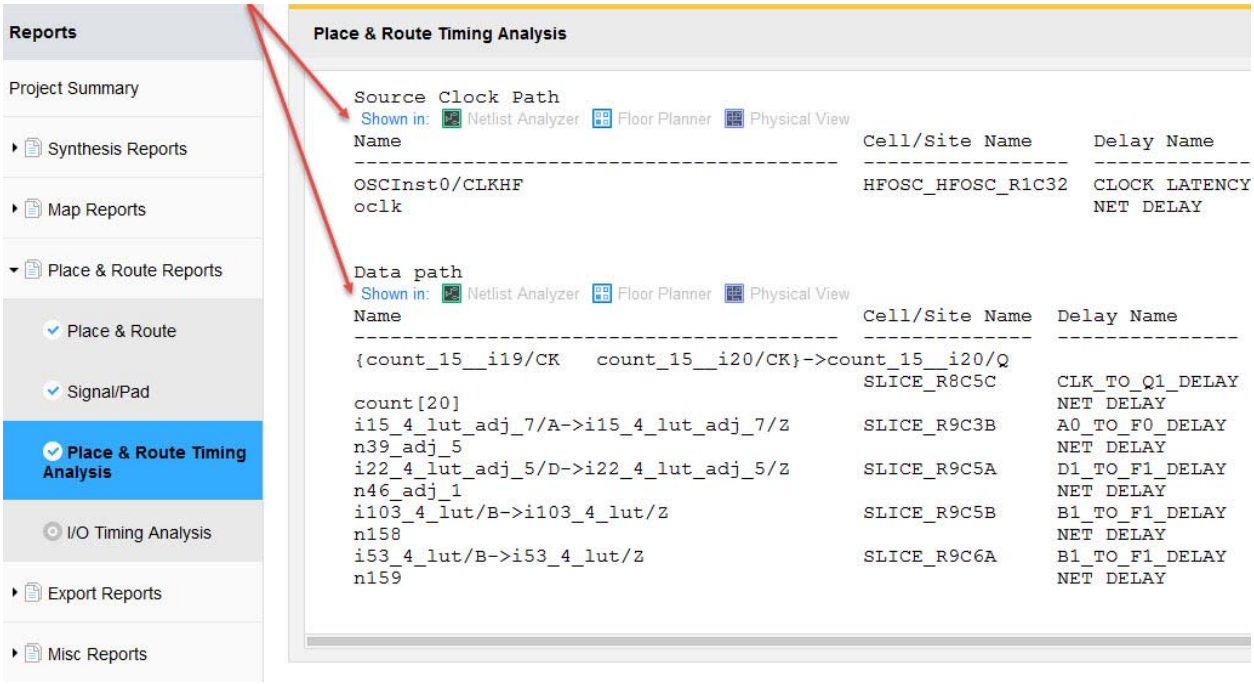
Figure 13: Radiant Software Timing Report



Upon clicking a link, a comprehensive report is shown detailing all calculations, as well as indicating whether a positive or negative slack has occurred.

The Reports view also supports a path cross-probing through the timing or analysis reports. You are able to view a specific clock or data path in Radiant software tools such as Netlist Analyzer, Floor Planner, or Physical View by clicking the icon next to the path. See Figure 14.

Figure 14: Cross-Probing from a Timing Report



A timing report consists of three primary sections:

- Paths for Various Constraint - Setup and Hold constraints based on SDC constraints applied.
- Critical Endpoint Summary - Detailed path summary with total slack, delay and clock information calculations based on endpoint element.
- Unconstrained Endpoints summary - A calculation resulting from no paths to an end point or because the end point was not properly constrained.

For more information on the details of Analyzing Static Timing, refer to the Radiant Help topic **User Guides > Analyze Static Timing > Running Timing Analysis > Timing Analysis Report File > Timing Analysis Report**.

Timing Analysis View

The redesigned Timing Analysis View features a detailed analysis of all constraints, as shown in Figure 15. To view path, path detail, and path calculations:

1. Click on a constraint in the Constraint box in the upper left.
2. Click on a path in the Path Summary box in the lower left.

The constraint's path, path detail and path calculations are shown on the right.

Figure 15: Timing Analysis View for the Radiant Software

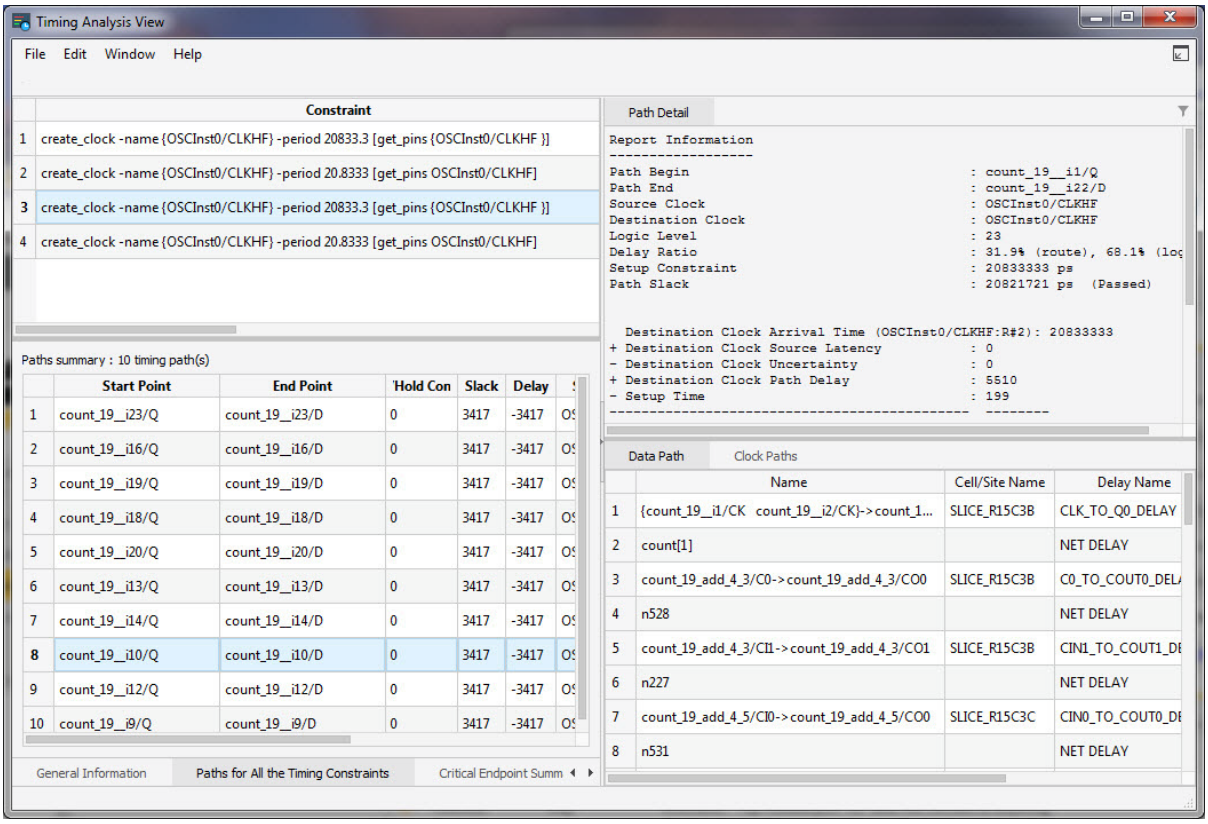
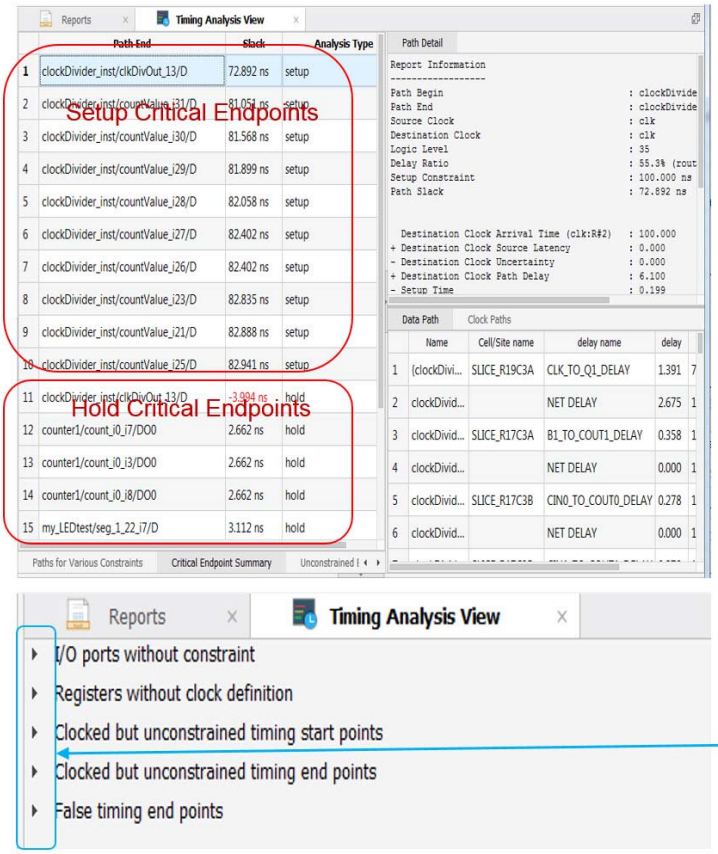


Figure 16 elaborates on Critical Endpoint Summary and Unconstrained Endpoint Summary, the two primary types of Timing Analysis available, that can assist you in debugging timing issues.

Figure 16: Types of Timing Analysis



- Critical Endpoint Summary
  - Reports worst slack paths on the top from the entire timing analysis
  - Number of paths user configurable (set to 10 shown)
  - Useful for a quick and bigger-picture TA result
- Unconstrained Endpoint Summary
  - Reported with 5 sub-sections
  - I/O, Registers, Clocks
  - False path endpoints are also reported here (Beta 5)
  - Expand each section to see the list of unconstrained endpoints

The Timing Analysis View also provides the ability to search elements and filter out specific paths for timing analysis. The following example shows how to analyze a clock domain crossing from clk_1Hz to clk:

1. Set the source (clk_1Hz).
2. Enter "**/Q**' in the filter box to identify start points.
3. Select all Q pins.
4. Move selection to the "From box >".
5. Enter "**/D**" in the filter box to get destination points.
6. Select all D pins.
7. Move selection to the "To box >".
   - All associated start and end points are listed in which a patch can be selected for analysis.
   - Path detail and calculation data are shown.
   - Apply the appropriate constraints for performance tuning.

Radiant Software Tools

The Radiant software provides an improved user friendly, concise, and efficient tool and design structure over its predecessor Lattice Diamond.

The following table lists the similarities and differences between Lattice Diamond and the Radiant software tools.

Table 52: Tools Comparison Between Lattice Diamond and the Radiant Software

Lattice Diamond Tools	Radiant Software Tools
Spreadsheet View Package View Device View Netlist View	Device Constraint Editor (DCE): Combines Spreadsheet, Package, Device and Netlist views into one tool for the primary purpose of entering physical (.pdc) constraints. This cross probing between multiple views. Some views in this tool can be hidden to maximize work space. For more information on Device Constraint Editor, see the Radiant software online Help topic **User Guides > Applying Design Constraints > Using Radiant Software Tools > Device Constraint Editor**.
Netlist Analyzer	Netlist Analyzer: Features are similar to Lattice Diamond.
NCD View	Not supported.
IPexpress	IP Catalog: Features are similar to Lattice Diamond, but with major enhancements added, such as IEEE 1735 encryption support for synthesis and simulation.
Reveal Inserter and Analyzer	Reveal Inserter and Analyzer: Features are similar to Lattice Diamond. The Power-On Reset (POR) Debug feature is disabled. A new Reveal Controller module enables access to a variety of registers and memory. For more information on the Reveal Controller Module, see the Radiant online Help topic **User Guides > Testing and Debugging On-Chip**.
Floorplan View	Floorplan View: Major features are similar to Lattice Diamond. Enhancements include World View and IO Planner features.

Table 52: Tools Comparison Between Lattice Diamond and the Radiant Software (Continued)

Lattice Diamond Tools	Radiant Software Tools
Physical View	Physical View: Allows you to easily locate target objects using logic hierarchy. Supports Netlist Widget, Property Widget, and World View. For IOLOGIC & I/O pads, the ASIC View replaces Logic Block View and Non-schematic table format. The Radiant Physical View supports cross probing between Floorplan View and Physical View and from DCE to Floorplan View. The Radiant Physical View also supports one-way probing for ports. The Radiant Physical View does not support switchbox and wire level display.
Timing Analysis View	Timing Analysis View: This tool has significant changes. For more information, refer to "Timing Analysis View" on page 75.
Power Calculator, integrated	Power Calculator, integrated and stand-alone: Features are similar to Lattice Diamond, plus Power Estimation mode in stand-alone.
Power Estimator, stand-alone	Power Estimator, stand-alone: Features are similar to Lattice Diamond.
Lattice Design Constraint (.LDC) Editor	Timing Constraint Editor: Pre-Synthesis Timing Constraint Editor tool used for generating or editing pre-synthesis timing constraints which are stored in .ldc file. Post-Synthesis Timing Constraint Editor Tool used for generating or editing post-synthesis timing constraints, stored in .pdc file. These constraints may override the pre-synthesis timing constraints. For more information, see "Radiant Constraint Tools" on page 52.
ECO Editor	ECO Editor: Features are similar to Lattice Diamond.
Programmer	Programmer: Features are similar to Lattice Diamond. Includes Programmer, Deployment Tool, Download Debugger, and Programming File Utility. Model 300 Programmer is not supported.
Partition Manager	Not supported.

Table 52: Tools Comparison Between Lattice Diamond and the Radiant Software (Continued)

Lattice Diamond Tools	Radiant Software Tools
Synplify Pro for Lattice	Synplify Pro for Lattice: Features are similar to Lattice Diamond. For more information on Synplify Pro for Lattice, from the Radiant software start page, click: **User Guides > Synopsys Synplify Pro for Lattice User Guide** and **User Guides > Synopsys Synplify Pro for Lattice Reference Manual**.
Active-HDL Lattice Edition	Active-HDL Lattice Edition (Windows only): Features are similar to Lattice Diamond. For more information about Active-HDL Lattice Edition, from the Radiant software start page, click: **User Guides > Active-HDL On-line Documentation (Windows only)**.
Run Manager	Run Manager: Features are similar to Lattice Diamond.
Simulation Wizard	Simulation Wizard: Features are similar to Lattice Diamond. Post-Synthesis simulation is supported.

Revision History

The following table gives the revision history for this document.

Date	Version	Description
05/09/2020	2.1	Updates to: ▶ "Changes to Arithmetic PMI for CrossLink-NX vs ECP5" on page 40. ▶ "Changes to Memory PMI for CrossLink-NX vs ECP5" on page 42.
11/21/2019	2.0	Added migrating ECP5 and CrossLink designs to CrossLink-NX.
03/25/2019	1.1	Added support for Radiant 1.1 software.
02/12/2018	1.0	Initial Release.