ECP5 and ECP5-5G sysDSP Usage Guide

Technical Note

FPGA-TN-02205-1.2

December 2020

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
DSP	Digital Signal Processing
FFT	Fast Fourier Transform
HDL	Hardware Description Language
RRM	Remote Radio Head

1. Introduction

This technical note discusses how to access the features of the ECP5™ and ECP5-5G™ sysDSP™ (Digital Signal Processing) slice described in ECP5 and ECP5-5G Family Data Sheet (FPGA-DS-02012). ECP5 and ECP5-5G devices are optimized to support high-performance DSP applications, such as wireless base station channel cards, Remote Radio Head (RRH) systems, video and imaging applications, and Fast Fourier Transform (FFT) functions.

2. sysDSP Overview

Figure 2.1 shows the diagram of the ECP5 and ECP5-5G DSP Block (two slices) at a higher level. As shown, each DSP slice has two 18-bit pre-adders, pre-adder registers, two 18-bit multipliers, input registers, pipeline registers, a 54-bit ALU, and output registers.

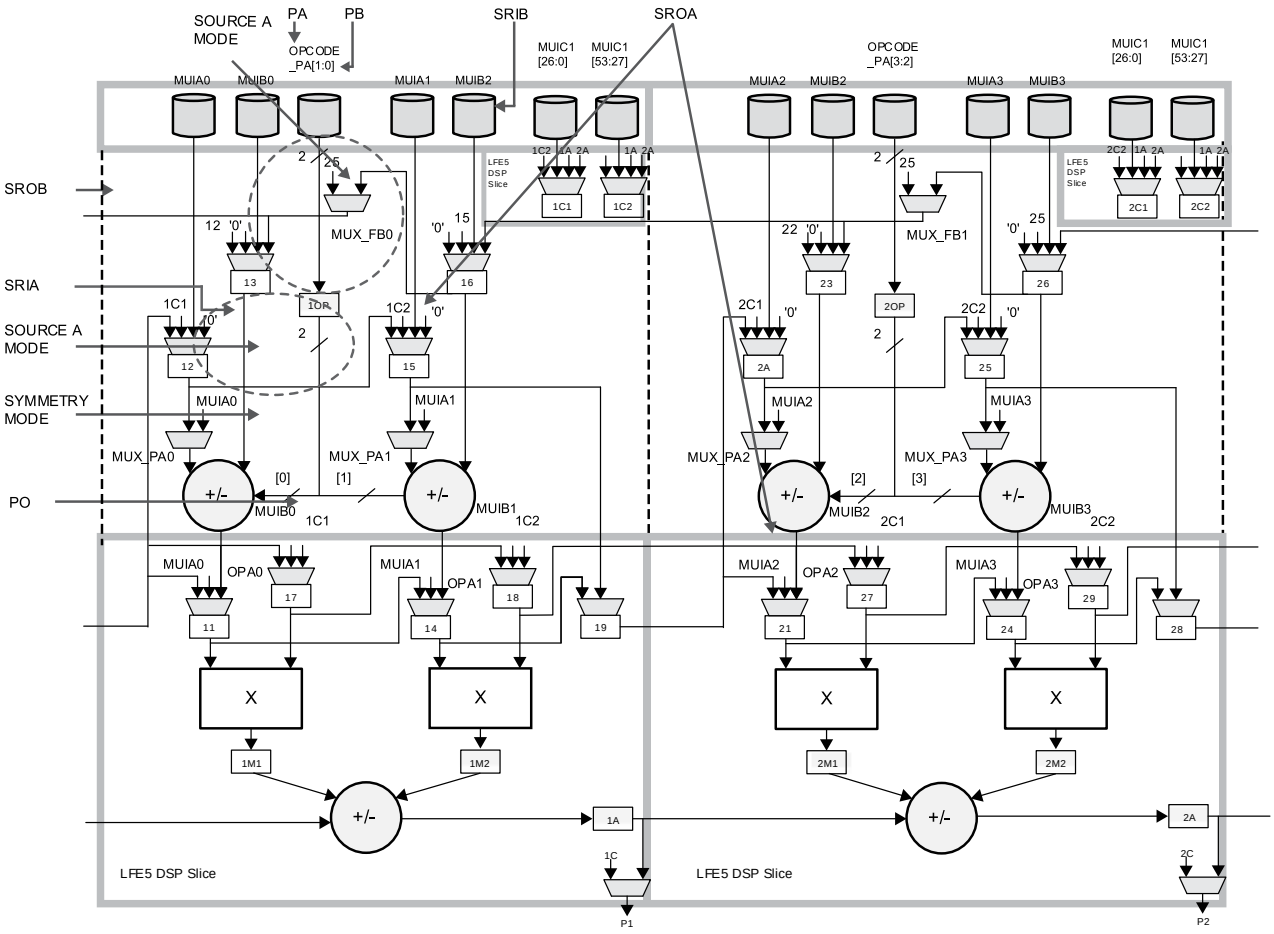

Figure 2.1. ECP5 and ECP5-5G DSP Block Diagram Overview

The sysDSP blocks are located in rows throughout the device. Figure 2.2 shows the block diagram of a single sysDSP slice. The programmable resources in a slice include the pre-adders, multipliers, ALU, multiplexers, pipeline registers, shift register chain and cascade chain. If the shift out register A is selected, the cascade match register (Casc) is available. The pre-adders and the multipliers can be configured as 9 bits or 18 bits wide and the ALU can be configured as 24 bits or 54 bits wide. Multipliers and accumulators can be configured independently and can be used as stand-alone primitives. However, pre-adders must only be used in conjunction with the associated multiplier block. Advanced features of the sysDSP slice are described later in this document.

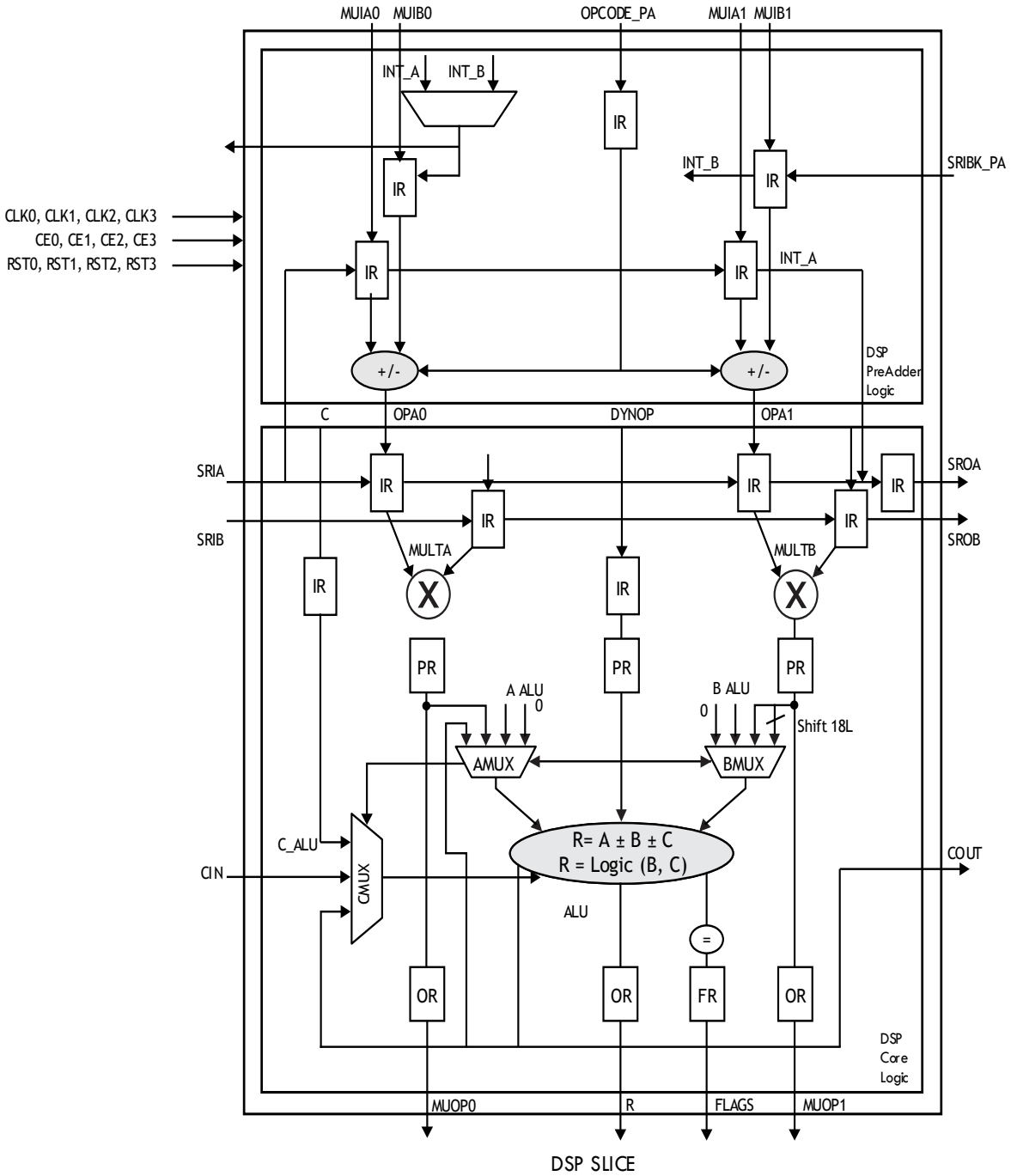

Figure 2.2. ECP5 and ECP5-5G DSP Slice Detailed View

Figure 2.2 shows the dual pre-adders in the core ECP5 and ECP5-5G DSP logic. The built-in pre-adders, multipliers, and ALU minimize the amount of external logic required to implement key DSP functions, resulting in efficient resource usage, reduced power consumption, and improved performance and data throughput for DSP applications. The ECP5 and ECP5-5G sysDSP slice can be configured several ways to suit your end application.

The various input, pipeline, and output registers in the DSP slice can be optionally used to improve operating frequency. There are four in total. The clock, clock enable, and reset connections to these registers are programmable in software. These can be set in IPexpress or during primitive or PMI instantiation. See the Attributes Description tables in the Targeting the sysDSP Slice by Instantiating Primitives section or Appendix A. Instantiating DSP Primitives in HDL for more details.

The input registers (IR) are 18-bits wide. The output registers (OR) and framing register (FR) share a 72-bit register. If simple multiplier mode is implemented, the register is used as a multiplier output register. If an ALU is implemented, it is used as the ALU output register. The simplified sysDSP block diagram is shown in Figure 2.3.

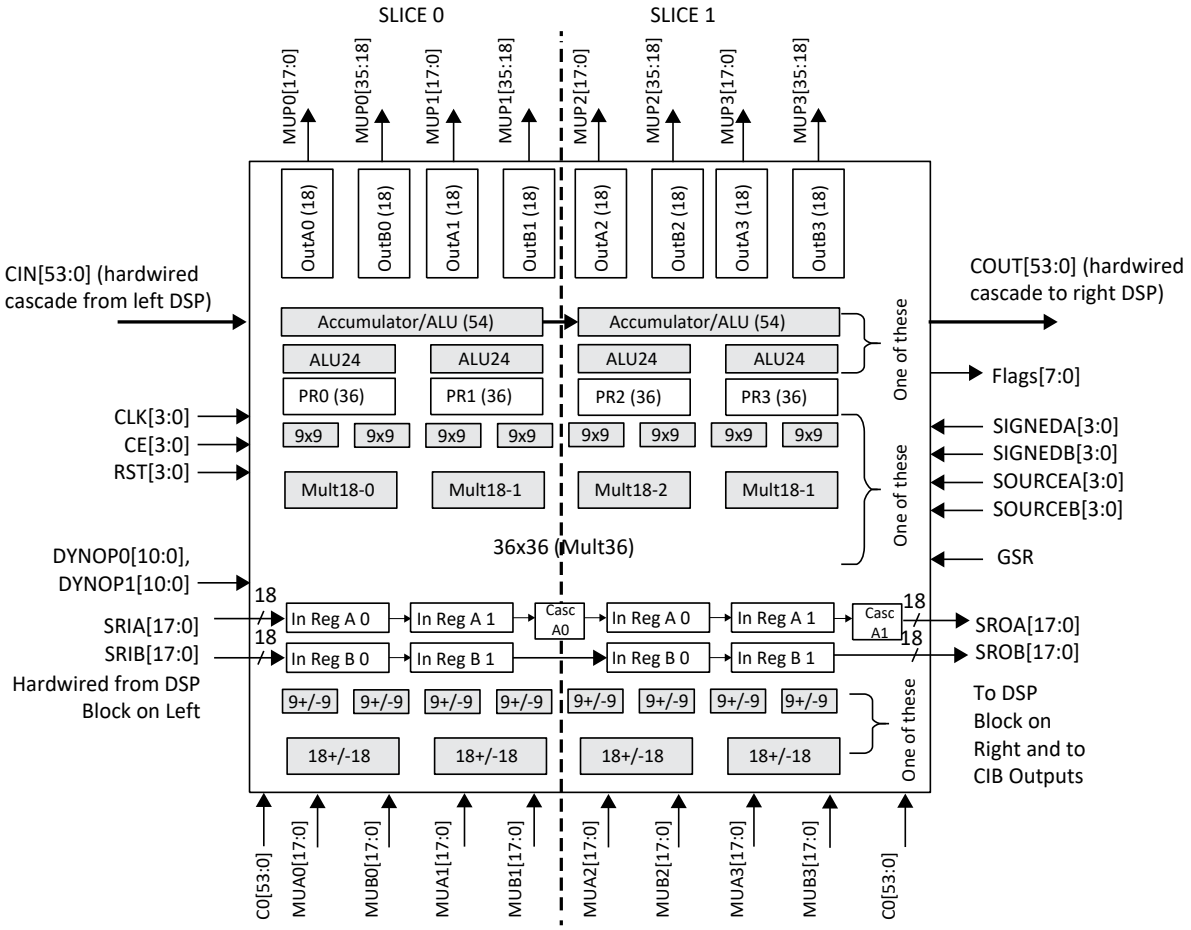

Figure 2.3. Simplified sysDSP Block Diagram

2.1. Operating Modes and Features

The DSP Block has three main operating modes:

- One 36 x 36 Multiplier
  - Basic Multiplier, no add/sub/accumulator/sum blocks.
- Four 18 x 18 Multipliers
  - Two add/sub/accumulator blocks
  - One summation Block for adding four multipliers
- Eight 9 x 9 Multipliers
  - Four add/sub/accumulator blocks
  - Two Summation Blocks

Additionally, the device has advanced features such as:

- 54-bit ternary adder/accumulator
- Additional multiplexer logic to support high-speed option
- Enhanced Pre-Adder Logic
  - 18-bit pre-adder/subtractor in front of each multiplier's sample register
  - Additional multiplexer logic to support high-speed option

In addition to these modes, ECP5 and ECP5-5G DSP Slice also includes pre-adders and additional shim logic to support:

- 1D Symmetry for Wireless Applications
- 2D Symmetry for Video Applications
- Long Tap FIR Filter Support across multiple DSP Rows
- Full 54-bit Accumulator Support

Various components are used in combination to enable the advanced functions of the sysDSP slice, such as:

- Cascading of slices for implementing adder trees in sysDSP slices
- Ternary addition functions implemented through the bypassing of multipliers
- Various rounding techniques that modify the data using the ALU
- ALU flags
- Dynamic multiplexer input selection allows for Time Division Multiplexing (TDM) of the sysDSP slice resources.
- High-speed logic to support the high-speed operating mode.

SOURCEA_MUX: SOURCEA_MUX selects between shift (SRIA) or parallel (A) input to the multiplier. SOURCEB_MUX: SOURCEB_MUX selects between shift (SRIB) or one of the parallel inputs (B or C).

AMUX: AMUX selects between multiple 54-bit inputs to the ALU statically or dynamically. The inputs to AMUX are listed in Table 4.1.

3. Using sysDSP

The DSP slices can be used in a number of ways in ECP5 and ECP5-5G devices, as described in the sections that follow.

3.1. Primitive Instantiation sysDSP

The sysDSP primitives can be directly instantiated in the design. Each of the primitives has a fixed set of attributes that can be customized to meet the design requirements.

An example of the primitive instantiation is provided in Appendix A. Instantiating DSP Primitives in HDL. You can get the detailed list of the primitives from the synthesis libraries under *cae_library\synthesis* folder under Diamond® installation.

3.2. Using Clarity Designer to Configure and Generate DSP Modules

Designers can utilize the Clarity Designer to easily specify a variety of DSP modules in their designs. Figure 3.1 shows a screenshot of the module selection for the memory modules under Clarity Designer in Lattice Diamond software.

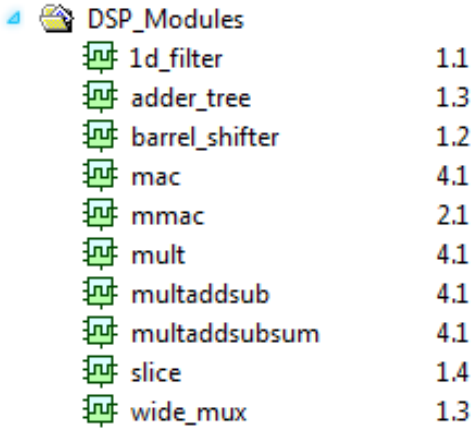

Figure 3.1. DSP Modules in Clarity Designer

3.2.1. Clarity Designer Flow

Clarity Designer allows you to generate, create (or open) any of the above modules for ECP5 and ECP5-5G devices.

From the Lattice Diamond software, select Tools > Clarity Designer.

Alternatively, you can also click on the  button in the toolbar. This opens the Clarity Designer window as shown in Figure 3.2.

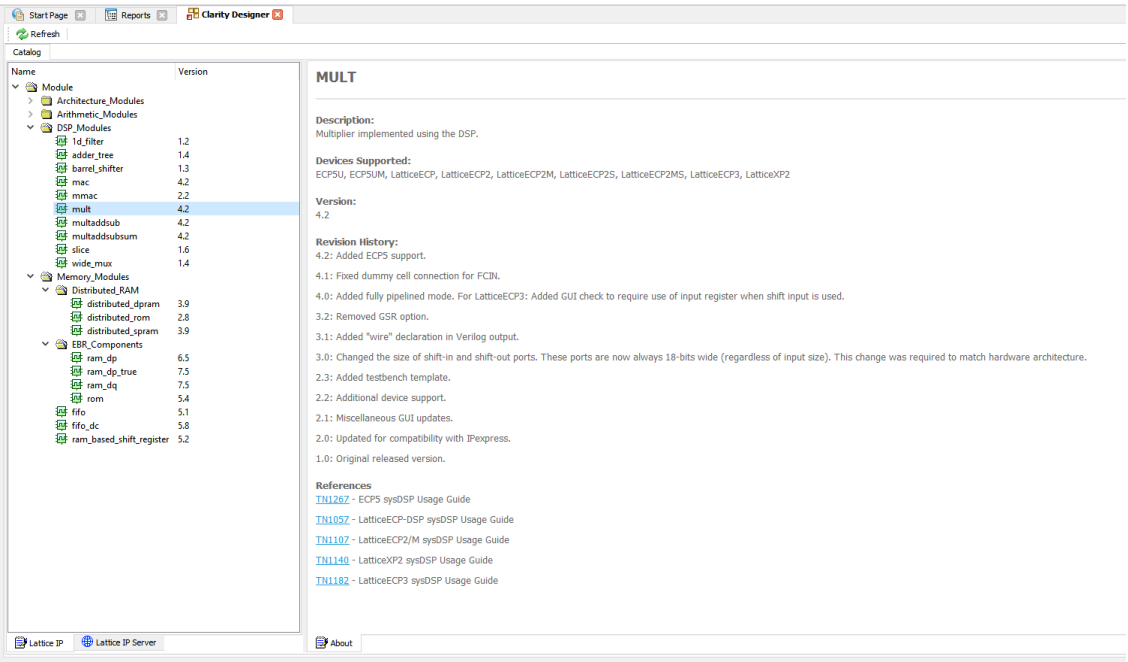

Figure 3.2. Clarity Designer in Lattice Diamond Software

The left section of Clarity Designer window has the Module tree, and all the sysDSP related modules are under DSP_Modules. The right section of the window provides a brief description of the selected module and links to further documentation.

Let us look at an example of generating an 18 x 18 multiplier using the Clarity Designer.

Double-click MULT under the DSP_Modules. This opens the Clarity Designer window that allows you to specify file name and macro name. Fill out the form, select the preferred language (Verilog or VHDL) and click Customize. Fill out the information of the module to generate. This is shown in Figure 3.3.

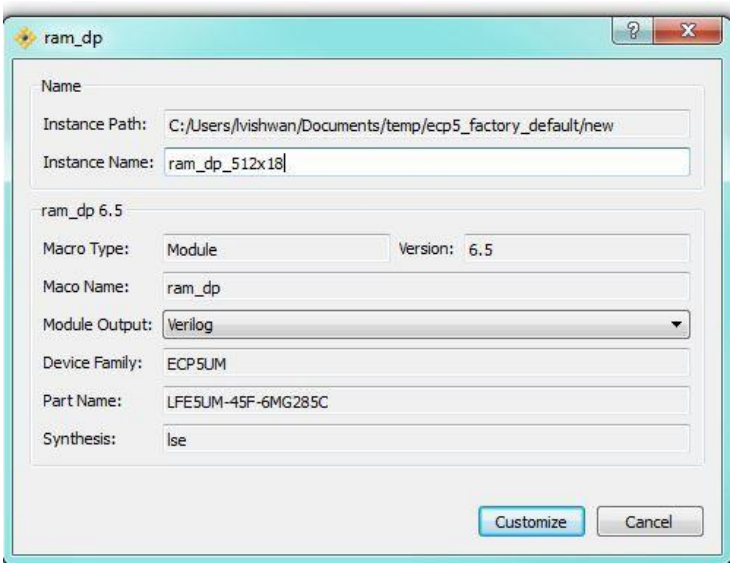

Figure 3.3. Generating Distributed 18x18 Multiplier in Clarity Designer in Lattice Diamond Software

Click Customize to open another window, as shown in Figure 3.4, where you can customize the 18 x 18 Multiplier.

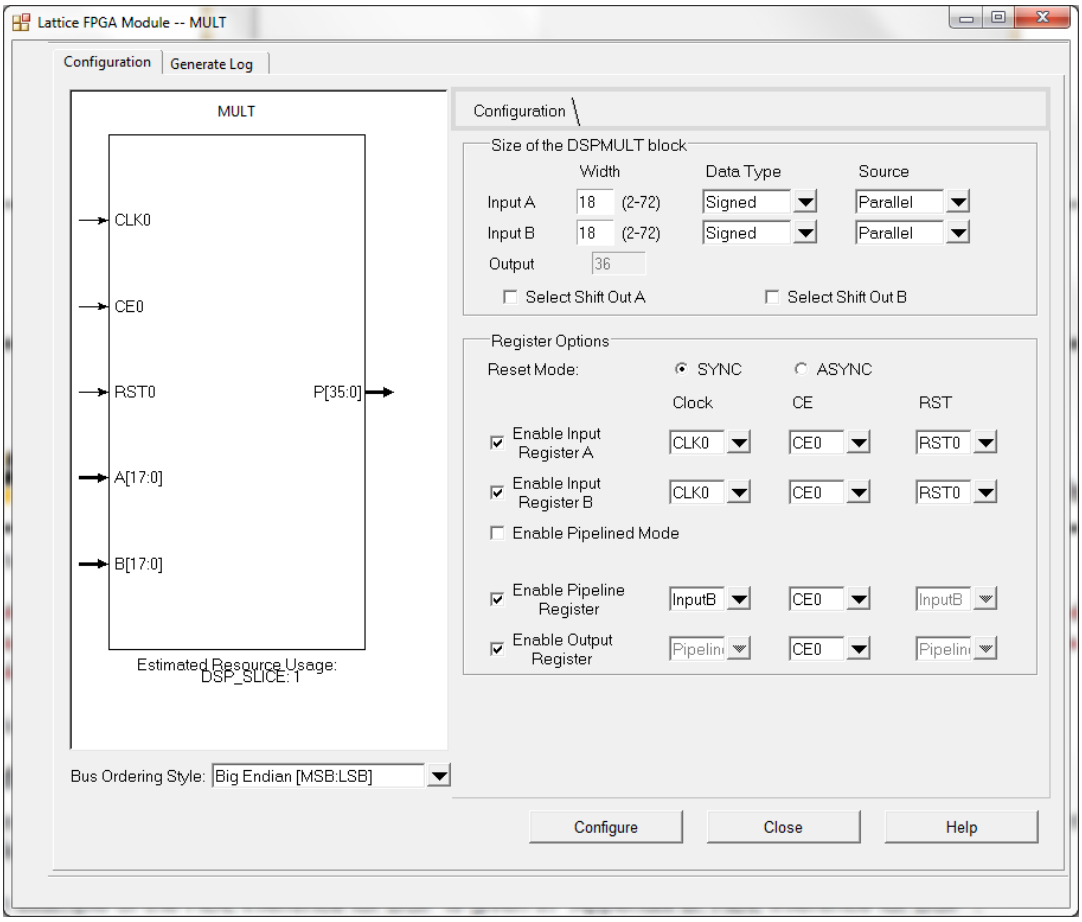

Figure 3.4. Customizing Multiplier in Clarity Designer in Lattice Diamond Software

Once all the right options of the module being generated are filled in, click on the Generate button. This module, once in the Diamond project, can be instantiated within other modules.

3.3. Inference sysDSP Slice

Designers can write a behavioral code for the DSP function such as multiplier, ALU, and the synthesis tool can infer the block into the ECP5 and ECP5-5G sysDSP functions.

An example of the HDL inference for DSP is provided in Appendix B. HDL Inference for DSP.

4. Targeting the sysDSP Slice by Instantiating Primitives

The sysDSP slice can be targeted by instantiating the sysDSP slice primitive into a design. The advantage of instantiating primitives is that it provides access to all the available ports and parameters. The disadvantage of this flow is that the customization requires additional coding and knowledge. This section details the primitives supported by ECP5 and ECP5-5G devices. See Appendix A. Instantiating DSP Primitives in HDL that shows an HDL example on how to instantiate sysDSP primitives.

The ECP5 and ECP5-5G sysDSP supports all the legacy ECP5 and ECP5-5G device primitives, namely MULT9X9C, MULT18X18C, ALU24A, and ALU24B. In addition, several other library primitives have been defined to take advantage of the features of the ECP5 and ECP5-5G sysDSP slice.

Various primitives available to the designers, along with the port definitions and attributes are discussed in the sections that follow.

4.1. MULT9X9C – Advanced 9X9 DSP Multiplier

The 9 x 9 multiplier is a widely used module. Figure 4.1 shows the MULT9X9C primitive available in the ECP5 and ECP5-5G device.

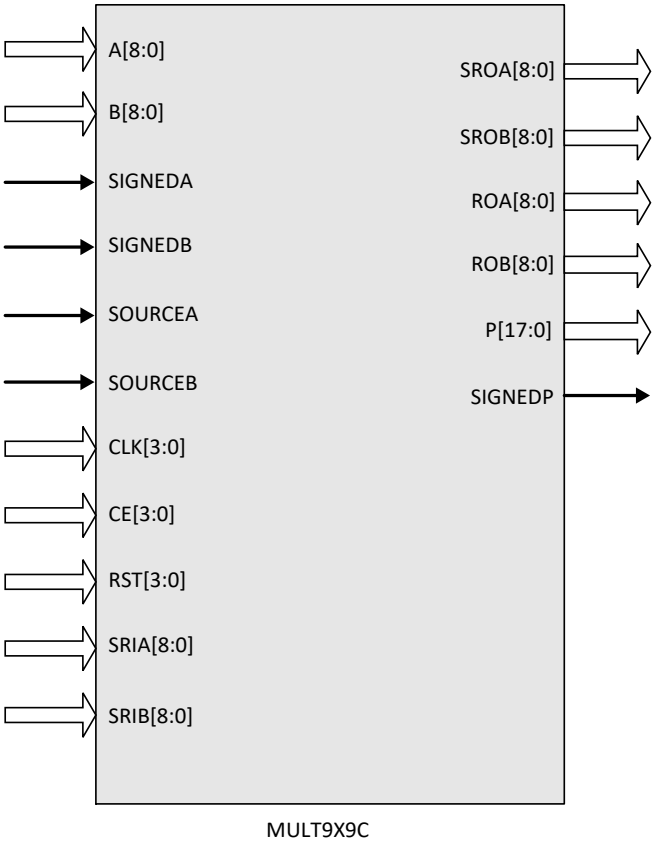

Figure 4.1. MULT9X9C Primitive

4.1.1. MULT9X9C – I/O Port Description

Table 4.1 describes the list of ports available for MULT9X9C primitive.

Table 4.1. MULT9X9C I/O Port Description

Port	Input/ Output	Description
A[8:0]	Input	Multiplier parallel Input A
B[8:0]	Input	Multiplier parallel Input B
SIGNEDA	Input	Signed Bit for Input A
SIGNEDB	Input	Signed Bit for Input B
SOURCEA	Input	Source Selector for Multiplier Input A
SOURCEB	Input	Source Selector for Multiplier Input B
CE[3:0]	Input	Clock Enable Inputs
CLK[3:0]	Input	Clock Inputs
RST[3:0]	Input	Reset Inputs
SRIA[8:0]	Input	Multiplier shift Input A
SRIB[8:0]	Input	Multiplier shift Input B
SROA[8:0]	Output	Shift Output A
SROB[8:0]	Output	Shift Output B
ROA[8:0]	Output	Output A
ROB[8:0]	Output	Output B
P[17:0]	Output	Product Output
SIGNEDP	Output	Signed Bit for the Product Output

4.1.2. MULT9X9C – Attribute Description

Table 4.2 describes the attributes for MULT9X9C primitive.

Table 4.2. Attribute Description for MULT9X9C

Attribute Name	Values	Default Value	User Interface Access
REG_INPUTA_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTA_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTA_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_INPUTB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTB_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTB_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_PIPELINE_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_PIPELINE_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_PIPELINE_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OUTPUT_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT_RST	RST0, RST1, RST2, RST3	RST0	Y
GSR	ENABLED, DISABLED	ENABLED	N
CAS_MATCH_REG	TRUE, FALSE	FALSE	Y
MULT_BYPASS	ENABLED, DISABLED	DISABLED	N
RESETMODE	SYNC, ASYNC	SYNC	Y

4.2. MULT18X18C – Basic 18X18 DSP Multiplier

The ECP5 and ECP5-5G device also includes the 18 X 18 multiplier natively. Figure 4.2 shows the MULT18X18C primitive available in ECP5 and ECP5-5G device.

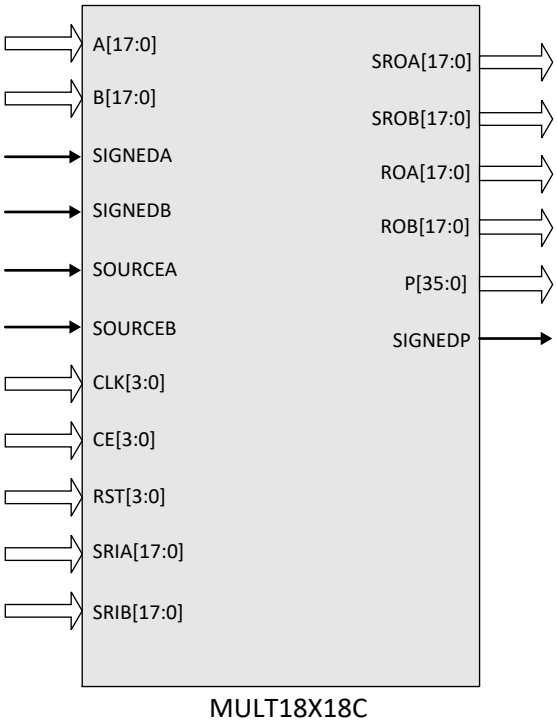

Figure 4.2. MULT18X18C Primitive

4.2.1. MULT18X18C – I/O Port Description

Table 4.3 describes the port list for MULT18X18C primitive.

Table 4.3. MULT18X18C I/O Port Description

Port	I/O	Description
A[17:0]	Input	Multiplier parallel Input A
B[17:0]	Input	Multiplier parallel Input B
SIGNEDA	Input	Signed Bit for Input A
SIGNEDB	Input	Signed Bit for Input B
SOURCEA	Input	Source Selector for Multiplier Input A
SOURCEB	Input	Source Selector for Multiplier Input B
CE[3:0]	Input	Clock Enable Inputs
CLK[3:0]	Input	Clock Inputs
RST[3:0]	Input	Reset Inputs
SRIA[17:0]	Input	Multiplier shift Input A
SRIB[17:0]	Input	Multiplier shift Input B
SROA[17:0]	Output	Shift Output A
SROB[17:0]	Output	Shift Output B
ROA[17:0]	Output	Output A
ROB[8:0]	Output	Output B
P[35:0]	Output	Product Output
SIGNEDP	Output	Signed Bit for the Product Output

4.2.2. MULT18X18C – Attribute Description

Table 4.4 describes the attributes for MULT18X18C primitive.

Table 4.4. Attribute Description for MULT18X18C

Attribute Name	Values	Default Value	User Interface Access
REG_INPUTA_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTA_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTA_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_INPUTB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTB_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTB_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_PIPELINE_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_PIPELINE_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_PIPELINE_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OUTPUT_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT_RST	RST0, RST1, RST2, RST3	RST0	Y
GSR	ENABLED, DISABLED	ENABLED	N
CAS_MATCH_REG	TRUE, FALSE	FALSE	Y
MULT_BYPASS	ENABLED, DISABLED	DISABLED	N
RESETMODE	SYNC, ASYNC	SYNC	Y

4.3. ALU24A – 24-bit Ternary Adder/ Subtractor

ECP5 and ECP5-5G devices also allows configuration in an ALU mode. Figure 4.3 shows the ALU24A primitive.

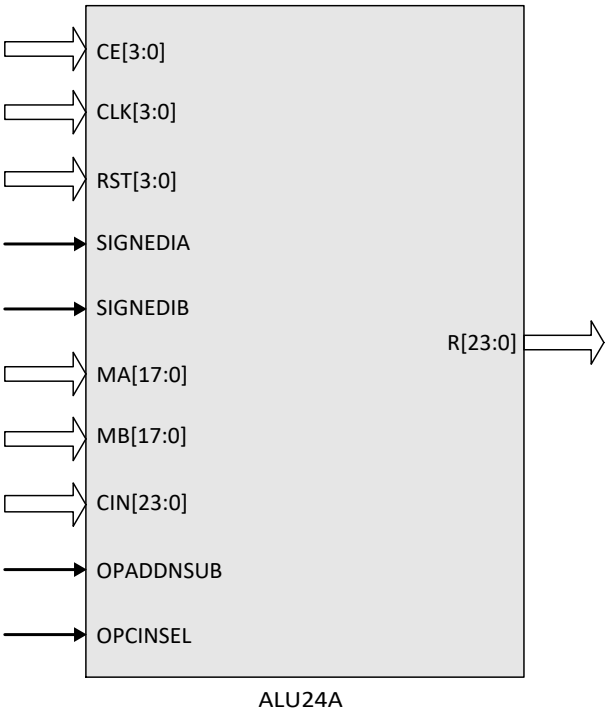

Figure 4.3. ALU24A Primitive

4.3.1. ALU24A – I/O Port Description

Table 4.5 describes the port list for ALU24A primitive.

Table 4.5. ALU24A I/O Port Description

Port	I/O	Description
CE[3:0]	Input	Clock Enable Inputs
CLK[3:0]	Input	Clock Inputs
RST[3:0]	Input	Reset Inputs
SIGNEDIA	Input	Sign Indicator for Input A
SIGNEDIB	Input	Sign Indicator for Input B
MA[17:0]	Input	Input A
MB[17:0]	Input	Input B
CIN[23:0]	Input	Carry In Input
OPADDNSUB	Input	Add/Sub Selector
OPCINSEL	Input	Carry In Selector
R[23:0]	Output	Sum Output

4.3.2. ALU24A – Attribute Description

Table 4.6 describes the attributes for ALU24A primitive.

Table 4.6. Attribute Description for ALU24A

Attribute Name	Values	Default Value	User Interface Access
REG_OUTPUT_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODE_0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODE_0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODE_0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODE_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODE_1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODE_1_RST	RST0, RST1, RST2, RST3	RST0	Y
GSR	ENABLED, DISABLED	ENABLED	N
RESETMODE	SYNC, ASYNC	SYNC	Y

4.4. ALU54A – 54-bit Ternary Adder/ Subtractor

Figure 4.4 shows the ALU54A primitive.

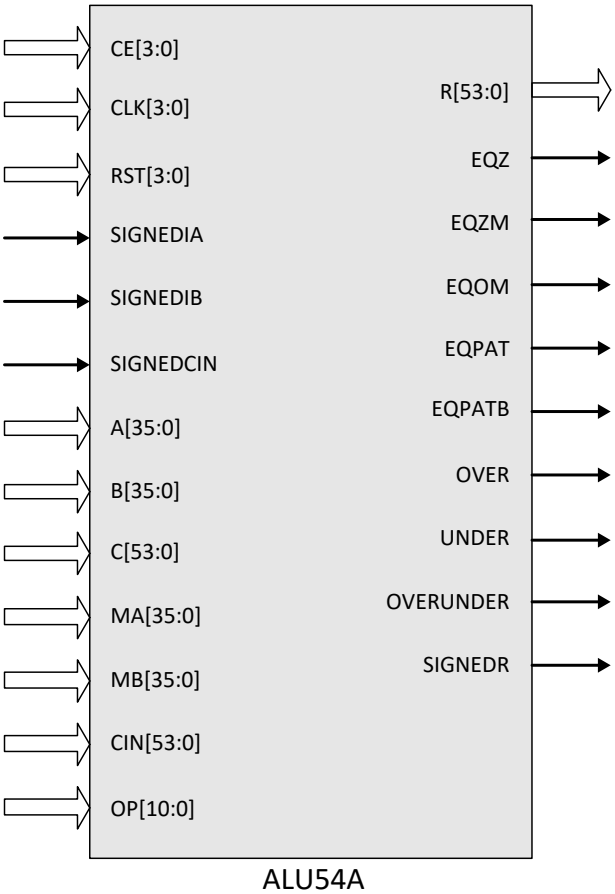

Figure 4.4. ALU54APrimitive

4.4.1. ALU24A – I/O Port Description

Table 4.7 describes the port list for ALU54A primitive.

Table 4.7. ALU54A I/O Port Description

Port	I/O	Description
CE[3:0]	Input	Clock Enable Inputs
CLK[3:0]	Input	Clock Inputs
RST[3:0]	Input	Reset Inputs
SIGNEDIA	Input	Sign Bit for Input A
SIGNEDIB	Input	Sign Bit for Input B
SIGNEDCIN	Input	Sign Bit for Carry In Input
A[35:0]	Input	Input A
B[35:0]	Input	Input B
C[53:0]	Input	Carry In Input
MA[35:0]	Input	Input A
MB[35:0]	Input	Input B
CIN[53:0]	Input	Carry In Input
OP[10:0]	Input	Opcode
R[53:0]	Output	Sum
EQZ	Output	Equal to Zero Flag
EQZM	Output	Equal to Zero with Mask Flag
EQOM	Output	Equal to One with Mask Flag
EQPAT	Output	Equal to Pattern with Mask Flag
EQPATB	Output	Equal to Bit Inverted Pattern with Mask Flag
OVER	Output	Accumulator Overflow
UNDER	Output	Accumulator Underflow
OVERUNDER	Output	Either Over on Underflow (may be removed)
SIGNEDR	Output	Sign Bit for Sum Output

4.4.2. ALU54A – Attribute Description

Table 4.8 describes the attributes for ALU54A primitive.

Table 4.8. Attribute Description for ALU54A

Attribute Name	Values	Default Value	User Interface Access
REG_INPUTC0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTC0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTC0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_INPUTC1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTC1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTC1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEOP0_0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEOP0_0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEOP0_0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEOP1_0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEOP0_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEOP0_1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEOP0_1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEOP1_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEIN_0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEIN_0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEIN_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEIN_1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEIN_1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OUTPUT0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OUTPUT1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_FLAG_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_FLAG_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_FLAG_RST	RST0, RST1, RST2, RST3	RST0	Y
MCPAT_SOURCE	STATIC, DYNAMIC	STATIC	Y
MASKPAT_SOURCE	STATIC, DYNAMIC	STATIC	Y
MASK01	0x00000000000000 to 0xFFFFFFFFFFFFF	0x00000000000000	Y
MCPAT	0x00000000000000 to 0xFFFFFFFFFFFFF	0x00000000000000	Y
MASKPAT	0x00000000000000 to 0xFFFFFFFFFFFFF	0x00000000000000	Y
RNDPAT	0x00000000000000 to 0xFFFFFFFFFFFFF	0x00000000000000	Y
GSR	ENABLED, DISABLED	ENABLED	N
RESETMODE	SYNC, ASYNC	SYNC	Y
MULT9_MODE	ENABLED, DISABLED	DISABLED	N
LEGACY	ENABLED, DISABLED	DISABLED	Y
FORCE_ZERO_BARREL_SHIFT	ENABLED, DISABLED	DISABLED	N

4.5. PRADD9A – 9-bit Pre-Adder/Shift

Figure 4.5 shows the PRADD9A primitive.

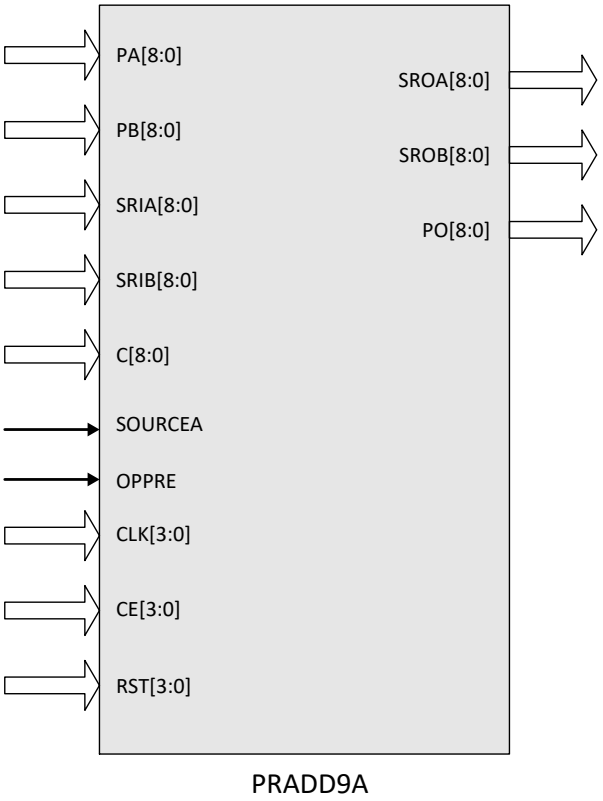

Figure 4.5. PRADD9A Primitive

4.5.1. PRADD9A – I/O Port Description

Table 4.9 describes the port list for PRADD9A primitive.

Table 4.9. PRADD9A I/O Port Description

Port	Tspec Port	I/O	Description
CE[3:0]	CE[3:0]	Input	Clock Enable Inputs
CLK[3:0]	CLK[3:0]	Input	Clock Inputs
RST[3:0]	RST[3:0]	Input	Reset Inputs
SOURCEA	SOURCEA	Input	Source Selector for Pre-adder Input A
PA[8:0]	MUA0/A1/A2/A3[8:0]	Input	Pre-adder Parallel Input A
PB[8:0]	MUB0/B1/B2/B3[8:0]	Input	Pre-adder Parallel Input B
SRIA[8:0]	SRIA[8:0]	Input	Pre-adder Shift Input A
SRIB[8:0]	SRI_PRE[8:0]	Input	Pre-adder Shift Input B, backward direction
C[8:0]	C[8:0]/C[35:27]	Input	Input used for high-speed option
SROA[8:0]	SROA[8:0]	Output	Pre-adder Shift Output A
SROB[8:0]	SRO_PRE[8:0]	Output	Pre-adder Shift Output B
PO[8:0]	OPA0	Output	Pre-adder Addition Output
OPPRE	OP_PRE	Input	Opcode for PreAdder

4.5.2. PRADD9A – Attribute Description

Table 4.10 describes the attributes for PRADD9A primitive.

Table 4.10. Attribute Description for PRADD9A

Attribute Name	Values	Default Value	User Interface Access	Tspec Name
REG_INPUTA_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	—
REG_INPUTA_CE	CE0, CE1, CE2, CE3	CE0	Y	—
REG_INPUTA_RST	RST0, RST1, RST2, RST3	RST0	Y	—
REG_INPUTB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	—
REG_INPUTB_CE	CE0, CE1, CE2, CE3	CE0	Y	—
REG_INPUTB_RST	RST0, RST1, RST2, RST3	RST0	Y	—
REG_INPUTC_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	—
REG_INPUTC_CE	CE0, CE1, CE2, CE3	CE0	Y	—
REG_INPUTC_RST	RST0, RST1, RST2, RST3	RST0	Y	—
REG_OPPRE_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	—
REG_OPPRE_CE	CE0, CE1, CE2, CE3	CE0	Y	—
REG_OPPRE_RST	RST0, RST1, RST2, RST3	RST0	Y	—
CLK0_DIV	ENABLED, DISABLED	ENABLED	Y	—
CLK1_DIV	ENABLED, DISABLED	ENABLED	Y	—
CLK2_DIV	ENABLED, DISABLED	ENABLED	Y	—
CLK3_DIV	ENABLED, DISABLED	ENABLED	Y	—
HIGHSPEED_CLK	NONE	NONE	Y	—
GSR	ENABLED, DISABLED	ENABLED	N	—
CAS_MATCH_REG	TRUE, FALSE	FALSE	Y	—
SOURCEA_MODE	A_SHIFT, C_SHIFT, A_C_DYNAMIC	A_SHIFT	Y	—
SOURCEB_MODE	SHIFT, PARALLEL, INTERNAL	SHIFT	Y	mc1_pa_b0
FB_MUX	SHIFT, SHIFT_BYPASS, DISABLED	SHIFT	Y	mc1_pa_fb
RESETMODE	SYNC, ASYNC	SYNC	Y	—
SYMMETRY_MODE	DIRECT, INTERNAL	DIRECT	Y	MUX_PA0/1/2/3

In the case of PRADD9A, you can also select the source for the input B. The details of SOURCEB_MODE Attribute for PRADD9A Primitive are given in Table 4.12. The Feedback Mux information is included in Table 4.13.

Table 4.11. SOURCEB_MODE Attribute for PRADD9A

IP Express Operation	SOURCEA_MODE Attribute	SOURCEA Port	Mc1_pa_mux3	Mc1_pa_mux4
Shift	A_SHIFT	1	00	01
A	A_SHIFT	0	00	00
C	C_SHIFT	0	01	00
A/C Dynamic	A_C_DYNAMIC	Live	10	00
Dynamic Shift/A	A_SHIFT	Live	00	10
Dynamic Shift/C	C_SHIFT	Live	01	10

Table 4.12. Details of SOURCEB_MODE Attribute for PRADD9A

SOURCEB_MODE Attribute	Operation (mc1_pa_b0 mux)
SHIFT	SRIB coming from the adjacent PREADDER on the right
PARALLEL	PB
INTERNAL	Output of Reg. 12

Table 4.13. Details of FB_MUX Attribute for PRADD9A

FB_MUX Attribute	Operation (MUX_FB0)
SHIFT	Output of Reg. 16
SHIFT_BYPASS	Output of Reg. 15
DISABLED	For placer only (PreAdder on the left side)

While using the PRADD9A primitive, it should be noted that each of the primitive can only drive PRADD9A in the adjacent column and/or MULT9X9D in the same column.

4.6. PRADD18A – 18-bit Pre-Adder/Shift

Figure 4.6 shows the PRADD18A primitive.

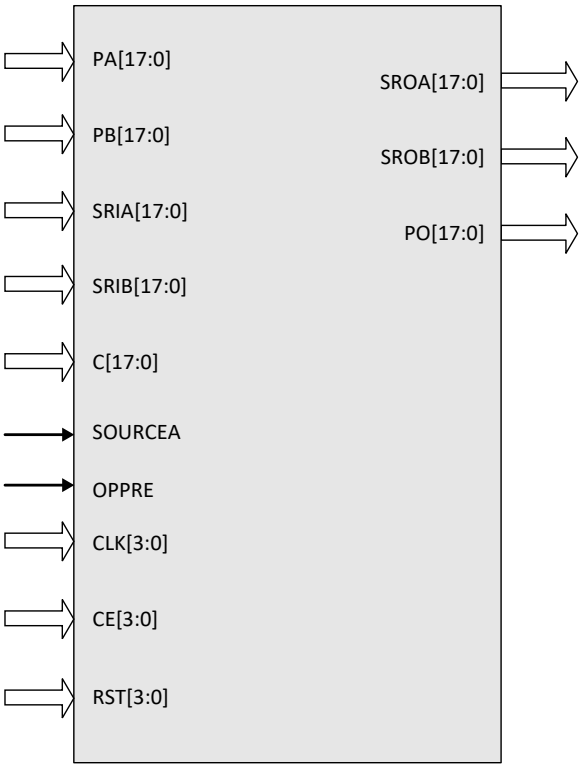

Figure 4.6. PRADD18A Primitive

4.6.1. PRADD18A – I/O Port Description

Table 4.14 describes the port list for PRADD18A primitive.

Table 4.14. PRADD18A I/O Port Description

Port	Tspec Port	I/O	Description
CE[3:0]	CE[3:0]	Input	Clock Enable Inputs
CLK[3:0]	CLK[3:0]	Input	Clock Inputs
RST[3:0]	RST[3:0]	Input	Reset Inputs
SOURCEA	SOURCEA_PRE[1:0]	Input	Source Selector for Pre-adder Input A
PA[17:0]	MUA0/A1/A2/A3[17:0]	Input	Pre-adder Parallel Input A
PB[17:0]	MUB0/B1/B2/B3[17:0]	Input	Pre-adder Parallel Input B
SRIA[17:0]	SRIA[17:0]	Input	Pre-adder Shift Input A
SRIB[17:0]	SRI_PRE[17:0]	Input	Pre-adder Shift Input A, backward direction
C[17:0]	C[17:0]/C[47:27]	Input	Input used for high-speed option
SROA[17:0]	SROA[17:0]	Output	Pre-adder Shift Output A
SROB[17:0]	SRO_PRE[17:0]	Output	Pre-adder Shift Output B
PO[17:0]	OPA0	Output	Pre-adder Addition Output
OPPRE	OP_PRE	Input	Opcode for PreAdder

4.6.2. PRADD18A – Attribute Description

Table 4.15 describes the attributes for PRADD18A primitive.

Table 4.15. Attribute Description for PRADD18A

Attribute Name	Values	Default Value	User Interface Access	Tspec Name
REG_INPUTA_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	—
REG_INPUTA_CE	CE0, CE1, CE2, CE3	CE0	Y	—
REG_INPUTA_RST	RST0, RST1, RST2, RST3	RST0	Y	—
REG_INPUTB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	—
REG_INPUTB_CE	CE0, CE1, CE2, CE3	CE0	Y	—
REG_INPUTB_RST	RST0, RST1, RST2, RST3	RST0	Y	—
REG_INPUTC_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	—
REG_INPUTC_CE	CE0, CE1, CE2, CE3	CE0	Y	—
REG_INPUTC_RST	RST0, RST1, RST2, RST3	RST0	Y	—
REG_OPPRE_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	—
REG_OPPRE_CE	CE0, CE1, CE2, CE3	CE0	Y	—
REG_OPPRE_RST	RST0, RST1, RST2, RST3	RST0	Y	—
CLK0_DIV	ENABLED, DISABLED	ENABLED	Y	—
CLK1_DIV	ENABLED, DISABLED	ENABLED	Y	—
CLK2_DIV	ENABLED, DISABLED	ENABLED	Y	—
CLK3_DIV	ENABLED, DISABLED	ENABLED	Y	—
HIGHSPEED_CLK	NONE	NONE	Y	—
GSR	ENABLED, DISABLED	ENABLED	N	—
CAS_MATCH_REG	TRUE, FALSE	FALSE	Y	—
SOURCEA_MODE	A_SHIFT, C_SHIFT, A_C_DYNAMIC,	A_SHIFT	Y	—
SOURCEB_MODE	SHIFT, PARALLEL, INTERNAL	SHIFT	Y	mc1_pa_b 0
FB_MUX	SHIFT, SHIFT_BYPASS, DISABLED	SHIFT	Y	mc1_pa_f b
RESETMODE	SYNC, ASYNC	SYNC	Y	—
PRADD_LOC	0, 1	0	Y	—
SYMMETRY_MODE	DIRECT, INTERNAL	DIRECT	Y	MUX_PA0/1/2/3

In case of PRADD18A, you can also select the source for the input B. The details of SOURCEB_MODE Attribute for PRADD18A Primitive, as given in Table 4.17. The Feedback Mux information is included in Table 4.18.

Table 4.16. SOURCEB_MODE Attribute for PRADD18A

Clarity Designer Operation	SOURCEA_MODE Attribute	SOURCEA Port	Mc1_pa_mux3	Mc1_pa_mux4
Shift	A_SHIFT	1	00	01
A	A_SHIFT	0	00	00
C	C_SHIFT	0	01	00
A/C Dynamic	A_C_DYNAMIC	Live	10	00
Dynamic Shift/A	A_SHIFT	Live	00	10
Dynamic Shift/C	C_SHIFT	Live	01	10

Table 4.17. Details of SOURCEB_MODE Attribute for PRADD18A

SOURCEB_MODE Attribute	Operation (mc1_pa_b0 mux)
SHIFT	SRIB coming from the adjacent PREADDER on the right
PARALLEL	PB
INTERNAL	Output of Reg. 12

Table 4.18. Details of FB_MUX Attribute for PRADD18A

FB_MUX Attribute	Operation (MUX_FB0)
SHIFT	Output of Reg. 16
SHIFT_BYPASS	Output of Reg. 15
DISABLED	For placer only (PreAdder on the left side)

While using the PRADD18A primitive, it should be noted that each of the primitive can only drive PRADD18A in the adjacent column and/or MULT18X18D in the same column.

Appendix A. Instantiating DSP Primitives in HDL

This appendix illustrates how to instantiate the ECP5 and ECP5-5G sysDSP primitives for both Verilog and VHDL.

Verilog Example Showing Snippet of the MULT18X18C Instantiation

```
defparam dsp_mult_0.MULT_BYPASS = "DISABLED" ;
defparam dsp_mult_0.CAS_MATCH_REG = "FALSE" ;
defparam dsp_mult_0.RESETMODE = "SYNC" ;
defparam dsp_mult_0.GSR = "ENABLED" ;
defparam dsp_mult_0.REG_OUTPUT_RST = "RST0" ;
defparam dsp_mult_0.REG_OUTPUT_CE = "CE0" ;
defparam dsp_mult_0.REG_OUTPUT_CLK = "NONE" ;
defparam dsp_mult_0.REG_PIPELINE_RST = "RST0" ;
defparam dsp_mult_0.REG_PIPELINE_CE = "CE0" ;
defparam dsp_mult_0.REG_PIPELINE_CLK = "CLK0" ;
defparam dsp_mult_0.REG_INPUTB_RST = "RST0" ;
defparam dsp_mult_0.REG_INPUTB_CE = "CE0" ;
defparam dsp_mult_0.REG_INPUTB_CLK = "CLK0" ;
defparam dsp_mult_0.REG_INPUTA_RST = "RST0" ;
defparam dsp_mult_0.REG_INPUTA_CE = "CE0" ;
defparam dsp_mult_0.REG_INPUTA_CLK = "CLK0" ;
MULT18X18C dsp_mult_0 (
    .A17(t5M1A_17), .A16(t5M1A_16), .A15(t5M1A_15), .A14(t5M1A_14),
    .A13(t5M1A_13), .A12(t5M1A_12), .A11(t5M1A_11), .A10(t5M1A_10),
    .A9(t5M1A_9), .A8(t5M1A_8), .A7(t5M1A_7), .A6(t5M1A_6), .A5(t5M1A_5),
    .A4(t5M1A_4), .A3(t5M1A_3), .A2(t5M1A_2), .A1(t5M1A_1),
    .A0(t5M1A_0), .B17(scuba_vlo), .B16(scuba_vlo), .B15(scuba_vlo),
    .B14(scuba_vlo), .B13(scuba_vlo), .B12(scuba_vlo), .B11(scuba_vlo),
    .B10(scuba_vlo), .B9(scuba_vlo), .B8(scuba_vlo), .B7(scuba_vlo),
    .B6(scuba_vlo), .B5(scuba_vlo), .B4(scuba_vlo), .B3(scuba_vlo),
    .B2(scuba_vlo), .B1(scuba_vlo), .B0(scuba_vlo), .SIGNEDA(scuba_vhi),
    .SIGNEDB(scuba_vhi), .SOURCEA(scuba_vlo), .SOURCEB(scuba_vlo),
    .CE0(ClockEn), .CE1(scuba_vlo), .CE2(scuba_vlo), .CE3(scuba_vlo),
    .CLK0(Clock), .CLK1(Clock_inv), .CLK2(scuba_vlo), .CLK3(scuba_vlo),
    .RST0(Reset), .RST1(scuba_vlo), .RST2(scuba_vlo), .RST3(scuba_vlo),
    .SRIA17(scuba_vlo), .SRIA16(scuba_vlo), .SRIA15(scuba_vlo), .SRIA14(scuba_vlo),
    .SRIA13(scuba_vlo), .SRIA12(scuba_vlo), .SRIA11(scuba_vlo),
    .SRIA10(scuba_vlo), .SRIA9(scuba_vlo), .SRIA8(scuba_vlo), .SRIA7(scuba_vlo),
    .SRIA6(scuba_vlo), .SRIA5(scuba_vlo), .SRIA4(scuba_vlo), .SRIA3(scuba_vlo),
    .SRIA2(scuba_vlo), .SRIA1(scuba_vlo), .SRIA0(scuba_vlo), .SRIB17(scuba_vlo),
    .SRIB16(scuba_vlo), .SRIB15(scuba_vlo), .SRIB14(scuba_vlo), .SRIB13(scuba_vlo),
    .SRIB12(scuba_vlo), .SRIB11(scuba_vlo), .SRIB10(scuba_vlo), .SRIB9(scuba_vlo),
    .SRIB8(scuba_vlo), .SRIB7(scuba_vlo), .SRIB6(scuba_vlo), .SRIB5(scuba_vlo),
    .SRIB4(scuba_vlo), .SRIB3(scuba_vlo), .SRIB2(scuba_vlo), .SRIB1(scuba_vlo),
    .SRIB0(scuba_vlo), .SROA17(), .SROA16(), .SROA15(), .SROA14(), .SROA13(),
    .SROA12(), .SROA11(), .SROA10(), .SROA9(), .SROA8(), .SROA7(), .SROA6(),
    .SROA5(), .SROA4(), .SROA3(), .SROA2(), .SROA1(), .SROA0(), .SROB17(),
    .SROB16(), .SROB15(), .SROB14(), .SROB13(), .SROB12(), .SROB11(),
    .SROB10(), .SROB9(), .SROB8(), .SROB7(), .SROB6(), .SROB5(), .SROB4(),
    .SROB3(), .SROB2(), .SROB1(), .SROB0(), .ROA17(roa1_5_17), .ROA16(roa1_5_16),
    .ROA15(roa1_5_15), .ROA14(roa1_5_14), .ROA13(roa1_5_13), .ROA12(roa1_5_12),
    .ROA11(roa1_5_11), .ROA10(roa1_5_10), .ROA9(roa1_5_9), .ROA8(roa1_5_8),
    .ROA7(roa1_5_7), .ROA6(roa1_5_6), .ROA5(roa1_5_5), .ROA4(roa1_5_4),
    .ROA3(roa1_5_3), .ROA2(roa1_5_2), .ROA1(roa1_5_1), .ROA0(roa1_5_0),
    .ROB17(rob1_5_17), .ROB16(rob1_5_16), .ROB15(rob1_5_15), .ROB14(rob1_5_14),
    .ROB13(rob1_5_13), .ROB12(rob1_5_12), .ROB11(rob1_5_11), .ROB10(rob1_5_10),
    .ROB9(rob1_5_9), .ROB8(rob1_5_8), .ROB7(rob1_5_7), .ROB6(rob1_5_6),
    .ROB5(rob1_5_5), .ROB4(rob1_5_4), .ROB3(rob1_5_3), .ROB2(rob1_5_2),
```

```
        .ROB1(rob1_5_1), .ROB0(rob1_5_0), .P35(t5P1_35), .P34(t5P1_34), .P33(t5P1_33),
        .P32(t5P1_32), .P31(t5P1_31), .P30(t5P1_30), .P29(t5P1_29), .P28(t5P1_28),
        .P27(t5P1_27), .P26(t5P1_26), .P25(t5P1_25), .P24(t5P1_24), .P23(t5P1_23),
        .P22(t5P1_22), .P21(t5P1_21), .P20(t5P1_20), .P19(t5P1_19), .P18(t5P1_18),
        .P17(t5P1_17), .P16(t5P1_16), .P15(t5P1_15), .P14(t5P1_14), .P13(t5P1_13),
        .P12(t5P1_12), .P11(t5P1_11), .P10(t5P1_10), .P9(t5P1_9), .P8(t5P1_8),
        .P7(t5P1_7), .P6(t5P1_6), .P5(t5P1_5), .P4(t5P1_4), .P3(t5P1_3), .P2(t5P1_2),
        .P1(t5P1_1), .P0(t5P1_0), .SIGNEDP(m5_signedp1)
        );
```

VHDL Example Showing Snippet of the ALU54A Instantiation

```
dsp_alu_0: ALU54A
   generic map (
        REG_OPCODEIN_1_RST=> "RST0", REG_OPCODEIN_1_CE=> "CE0",
        REG_OPCODEIN_1_CLK=> "NONE", REG_OPCODEIN_0_RST=> "RST0",
        REG_OPCODEIN_0_CE=> "CE0", REG_OPCODEIN_0_CLK=> "NONE",
        REG_OPCODEOP1_1_CLK=> "NONE", REG_OPCODEOP1_0_CLK=> "NONE",
        REG_OPCODEOP0_1_RST=> "RST0", REG_OPCODEOP0_1_CE=> "CE0",
        REG_OPCODEOP0_1_CLK=> "NONE", REG_OPCODEOP0_0_RST=> "RST0",
        REG_OPCODEOP0_0_CE=> "CE0", REG_OPCODEOP0_0_CLK=> "NONE",
        REG_INPUTC1_RST=> "RST0", REG_INPUTC1_CE=> "CE0",
        REG_INPUTC1_CLK=> "NONE", REG_INPUTC0_RST=> "RST0",
        REG_INPUTC0_CE=> "CE0", REG_INPUTC0_CLK=> "NONE", LEGACY=> "DISABLED",
        REG_FLAG_RST=> "RST0", REG_FLAG_CE=> "CE0", REG_FLAG_CLK=> "NONE",
        REG_OUTPUT1_RST=> "RST0", REG_OUTPUT1_CE=> "CE0",
        REG_OUTPUT1_CLK=> "CLK0", REG_OUTPUT0_RST=> "RST0",
        REG_OUTPUT0_CE=> "CE0", REG_OUTPUT0_CLK=> "CLK0", MULT9_MODE=> "DISABLED",
        RNDPAT=> "0x00000000000000", MASKPAT=> "0x00000000000000", MCPAT=>
        "0x00000000000000",
        MASK01=> "0x00000000000000", MASKPAT_SOURCE=> "STATIC",
        MCPAT_SOURCE=> "STATIC", RESETMODE=> "SYNC", GSR=> "ENABLED"
        )

    port map (
        A35=>rob0_5_17, A34=>rob0_5_16, A33=>rob0_5_15,
        A32=>rob0_5_14, A31=>rob0_5_13, A30=>rob0_5_12,
        A29=>rob0_5_11, A28=>rob0_5_10, A27=>rob0_5_9, A26=>rob0_5_8,
        A25=>rob0_5_7, A24=>rob0_5_6, A23=>rob0_5_5, A22=>rob0_5_4,
        A21=>rob0_5_3, A20=>rob0_5_2, A19=>rob0_5_1, A18=>rob0_5_0,
        A17=>roa0_5_17, A16=>roa0_5_16, A15=>roa0_5_15,
        A14=>roa0_5_14, A13=>roa0_5_13, A12=>roa0_5_12,
        A11=>roa0_5_11, A10=>roa0_5_10, A9=>roa0_5_9, A8=>roa0_5_8,
        A7=>roa0_5_7, A6=>roa0_5_6, A5=>roa0_5_5, A4=>roa0_5_4,
        A3=>roa0_5_3, A2=>roa0_5_2, A1=>roa0_5_1, A0=>roa0_5_0,
        B35=>rob1_5_17, B34=>rob1_5_16, B33=>rob1_5_15,
        B32=>rob1_5_14, B31=>rob1_5_13, B30=>rob1_5_12,
        B29=>rob1_5_11, B28=>rob1_5_10, B27=>rob1_5_9, B26=>rob1_5_8,
        B25=>rob1_5_7, B24=>rob1_5_6, B23=>rob1_5_5, B22=>rob1_5_4,
        B21=>rob1_5_3, B20=>rob1_5_2, B19=>rob1_5_1, B18=>rob1_5_0,
        B17=>roa1_5_17, B16=>roa1_5_16, B15=>roa1_5_15,
        B14=>roa1_5_14, B13=>roa1_5_13, B12=>roa1_5_12,
        B11=>roa1_5_11, B10=>roa1_5_10, B9=>roa1_5_9, B8=>roa1_5_8,
        B7=>roa1_5_7, B6=>roa1_5_6, B5=>roa1_5_5, B4=>roa1_5_4,
        B3=>roa1_5_3, B2=>roa1_5_2, B1=>roa1_5_1, B0=>roa1_5_0,
        C53=>scuba_vlo, C52=>scuba_vlo, C51=>scuba_vlo,
        C50=>scuba_vlo, C49=>scuba_vlo, C48=>scuba_vlo,
        C47=>scuba_vlo, C46=>scuba_vlo, C45=>scuba_vlo,
        C44=>scuba_vlo, C43=>scuba_vlo, C42=>scuba_vlo,
```

```
        C41=>scuba_vlo, C40=>scuba_vlo, C39=>scuba_vlo,
        C38=>scuba_vlo, C37=>scuba_vlo, C36=>scuba_vlo,
        C35=>scuba_vlo, C34=>scuba_vlo, C33=>scuba_vlo,
        C32=>scuba_vlo, C31=>scuba_vlo, C30=>scuba_vlo,
        C29=>scuba_vlo, C28=>scuba_vlo, C27=>scuba_vlo,
        C26=>scuba_vlo, C25=>scuba_vlo, C24=>scuba_vlo,
        C23=>scuba_vlo, C22=>scuba_vlo, C21=>scuba_vlo,
        C20=>scuba_vlo, C19=>scuba_vlo, C18=>scuba_vlo,
        C17=>scuba_vlo, C16=>scuba_vlo, C15=>scuba_vlo,
        C14=>scuba_vlo, C13=>scuba_vlo, C12=>scuba_vlo,
        C11=>scuba_vlo, C10=>scuba_vlo, C9=>scuba_vlo, C8=>scuba_vlo,
        C7=>scuba_vlo, C6=>scuba_vlo, C5=>scuba_vlo, C4=>scuba_vlo,
        C3=>scuba_vlo, C2=>scuba_vlo, C1=>scuba_vlo, C0=>scuba_vlo,
        CE0=>ClockEn, CE1=>scuba_vlo, CE2=>scuba_vlo, CE3=>scuba_vlo,
        CLK0=>Clock, CLK1=>Clock_inv, CLK2=>scuba_vlo,
        CLK3=>scuba_vlo, RST0=>Reset, RST1=>scuba_vlo,
        RST2=>scuba_vlo, RST3=>scuba_vlo, SIGNEDIA=>m5_signedp0,
        SIGNEDIB=>m5_signedp1, SIGNEDCIN=>signr4, MA35=>t5P0_35,
        MA34=>t5P0_34, MA33=>t5P0_33, MA32=>t5P0_32, MA31=>t5P0_31,
        MA30=>t5P0_30, MA29=>t5P0_29, MA28=>t5P0_28, MA27=>t5P0_27,
        MA26=>t5P0_26, MA25=>t5P0_25, MA24=>t5P0_24, MA23=>t5P0_23,
        MA22=>t5P0_22, MA21=>t5P0_21, MA20=>t5P0_20, MA19=>t5P0_19,
        MA18=>t5P0_18, MA17=>t5P0_17, MA16=>t5P0_16, MA15=>t5P0_15,
        MA14=>t5P0_14, MA13=>t5P0_13, MA12=>t5P0_12, MA11=>t5P0_11,
        MA10=>t5P0_10, MA9=>t5P0_9, MA8=>t5P0_8, MA7=>t5P0_7,
        MA6=>t5P0_6, MA5=>t5P0_5, MA4=>t5P0_4, MA3=>t5P0_3,
        MA2=>t5P0_2, MA1=>t5P0_1, MA0=>t5P0_0, MB35=>t5P1_35,
        MB34=>t5P1_34, MB33=>t5P1_33, MB32=>t5P1_32, MB31=>t5P1_31,
        MB30=>t5P1_30, MB29=>t5P1_29, MB28=>t5P1_28, MB27=>t5P1_27,
        MB26=>t5P1_26, MB25=>t5P1_25, MB24=>t5P1_24, MB23=>t5P1_23,
        MB22=>t5P1_22, MB21=>t5P1_21, MB20=>t5P1_20, MB19=>t5P1_19,
        MB18=>t5P1_18, MB17=>t5P1_17, MB16=>t5P1_16, MB15=>t5P1_15,
        MB14=>t5P1_14, MB13=>t5P1_13, MB12=>t5P1_12, MB11=>t5P1_11,
        MB10=>t5P1_10, MB9=>t5P1_9, MB8=>t5P1_8, MB7=>t5P1_7,
        MB6=>t5P1_6, MB5=>t5P1_5, MB4=>t5P1_4, MB3=>t5P1_3,
        MB2=>t5P1_2, MB1=>t5P1_1, MB0=>t5P1_0, CIN53=>r4_53,
        CIN52=>r4_52, CIN51=>r4_51, CIN50=>r4_50, CIN49=>r4_49,
        CIN48=>r4_48, CIN47=>r4_47, CIN46=>r4_46, CIN45=>r4_45,
        CIN44=>r4_44, CIN43=>r4_43, CIN42=>r4_42, CIN41=>r4_41,
        CIN40=>r4_40, CIN39=>r4_39, CIN38=>r4_38, CIN37=>r4_37,
        CIN36=>r4_36, CIN35=>r4_35, CIN34=>r4_34, CIN33=>r4_33,
        CIN32=>r4_32, CIN31=>r4_31, CIN30=>r4_30, CIN29=>r4_29,
        CIN28=>r4_28, CIN27=>r4_27, CIN26=>r4_26, CIN25=>r4_25,
        CIN24=>r4_24, CIN23=>r4_23, CIN22=>r4_22, CIN21=>r4_21,
        CIN20=>r4_20, CIN19=>r4_19, CIN18=>r4_18, CIN17=>r4_17,
        CIN16=>r4_16, CIN15=>r4_15, CIN14=>r4_14, CIN13=>r4_13,
        CIN12=>r4_12, CIN11=>r4_11, CIN10=>r4_10, CIN9=>r4_9,
        CIN8=>r4_8, CIN7=>r4_7, CIN6=>r4_6, CIN5=>r4_5, CIN4=>r4_4,
        CIN3=>r4_3, CIN2=>r4_2, CIN1=>r4_1, CIN0=>r4_0,
        OP10=>scuba_vlo, OP9=>scuba_vhi, OP8=>scuba_vlo,
        OP7=>scuba_vlo, OP6=>scuba_vlo, OP5=>scuba_vhi,
        OP4=>scuba_vlo, OP3=>scuba_vhi, OP2=>scuba_vhi,
        OP1=>scuba_vhi, OP0=>scuba_vhi, R53=>r5_53, R52=>r5_52,
        R51=>r5_51, R50=>r5_50, R49=>r5_49, R48=>r5_48, R47=>r5_47,
        R46=>r5_46, R45=>r5_45, R44=>r5_44, R43=>r5_43, R42=>r5_42,
        R41=>r5_41, R40=>r5_40, R39=>r5_39, R38=>r5_38, R37=>r5_37,
        R36=>r5_36, R35=>r5_35, R34=>r5_34, R33=>r5_33, R32=>r5_32,
        R31=>r5_31, R30=>r5_30, R29=>r5_29, R28=>r5_28, R27=>r5_27,
        R26=>r5_26, R25=>r5_25, R24=>r5_24, R23=>r5_23, R22=>r5_22,
```

```
        R21=>r5_21, R20=>r5_20, R19=>r5_19, R18=>r5_18, R17=>r5_17,
        R16=>r5_16, R15=>r5_15, R14=>r5_14, R13=>r5_13, R12=>r5_12,
        R11=>r5_11, R10=>r5_10, R9=>r5_9, R8=>r5_8, R7=>r5_7,
        R6=>r5_6, R5=>r5_5, R4=>r5_4, R3=>r5_3, R2=>r5_2, R1=>r5_1,
        R0=>r5_0, EQZ=>open, EQZM=>open, EQOM=>open,
        EQPAT=>open, EQPATB=>open, OVER=>open, UNDER=>open,
        OVERUNDER=>open, SIGNEDR=>signr5
        );
```

Appendix B. HDL Inference for DSP

Synthesis inference flow enables the design tools to infer sysDSP slices from an HDL design. It is important to note that when using the inference flow, unless the code style matches the sysDSP slice, results are not optimal. You can infer the ECP5 and ECP5-5G sysDSP slice with Synplify Pro® from Synopsys or the Lattice Synthesis Engine (LSE) if certain coding guidelines are followed. The following are VHDL and Verilog examples. This example would not have functional simulation support. This is for example purposes only.

VHDL Example to Infer Fully Pipelined Multiplier

```
library ieee;
use ieee.std_logic_1164.all;
--use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity mult is
        port (reset, clk : in std_logic;
        dataax, dataay : in std_logic_vector(8 downto 0);
        dataout : out std_logic_vector (17 downto 0));
end;

architecture arch of mult is
        signal dataax_reg, dataay_reg : std_logic_vector (8 downto 0);
        signal dataout_node : std_logic_vector (17 downto 0);
        signal dataout_pipeline : std_logic_vector (17 downto 0);
begin
        process (clk, reset)
begin
        if (reset='1') then
        dataax_reg <= (others => '0');
        dataay_reg <= (others => '0');
        elsif (clk'event and clk='1') then
        dataax_reg <= dataax;
        dataay_reg <= dataay;
        end if;
        end process;

    dataout_node <= dataax_reg * dataay_reg;
        process (clk, reset)
        begin
        if (reset='1') then
        dataout_pipeline <= (others => '0');
        elsif (clk'event and clk='1') then
        dataout_pipeline <= dataout_node;
        end if;
end process;

process (clk, reset)
begin
        if (reset='1') then
        dataout <= (others => '0');
        elsif (clk'event and clk='1') then
        dataout <= dataout_pipeline;
        end if;
        end process;
end arch;•
```

Verilog Example to Infer Fully Pipelined Multiplier

```
module mult (dataout, dataax, dataay, clk, reset);
output [35:0] dataout;
input [17:0] dataax, dataay;
input clk,reset;

reg [35:0] dataout;
reg [17:0] dataax_reg, dataay_reg;
wire [35:0] dataout_node;
reg [35:0] dataout_reg;

always @(posedge clk or posedge reset)
    begin
        if (reset)
        begin
        dataax_reg <= 0;
        dataay_reg <= 0;
        end
        else
        begin
        dataax_reg <= dataax;
        dataay_reg <= dataay;
        end
    end

assign dataout_node = dataax_reg * dataay_reg;
always @(posedge clk or posedge reset)
    begin
        if (reset)
        dataout_reg <= 0;
        else
        dataout_reg <= dataout_node;
        end

always @(posedge clk or posedge reset)
        begin
        if (reset)
        dataout <= 0;
        else
        dataout <= dataout_reg;
    end
endmodule
```

Technical Support Assistance

Submit a technical support case via www.latticesemi.com/techsupport.

Revision History

Revision 1.2, December 2020

Section	Change Summary
All	• Changed document number from TN1267 to FPGA-TN-02205. • Updated document template.
Disclaimers	Added this section.
Acronyms in This Document	Added this section.
sysDSP Overview	• Updated the paragraphs in sysDSP Overview section to remove any reference to Diamond Help. • Updated Figure 2.1 and Figure 2.2. • Added Figure 2.3. • Removed *Higher Operation of Frequency (400 MHz)* and *18-bit dual multipliers* support in Operating Modes and Features.
Using sysDSP	Updated Figure 3.2.

Targeting the sysDSP Slice by Instantiating Primitives	• Removed the following sections:
	  • MULT9X9D – Advanced 9X9 DSP Multiplier for Highspeed
	  • MULT18X18D – Advanced 18X18 DSP Multiplier for High Speed
	  • ALU24B – 24-bit Ternary Adder/Subtractor for 9X9 Mode
	  • ALU54B – 54-bit Ternary Adder/Subtractor for High Speed
	• Updated *HIGHSPEED_CLK* and *SOURCEA_MODE* values in Table 4.10 and Table 4.15.
	• Removed *HighspeedAC* from Table 4.11 and Table 4.16.

Appendix A. Instantiating DSP Primitives in HDL	Updated Verilog and VHDL examples.

Revision 1.1, November 2015

Section	Change Summary
All	• Added support for ECP5-5G. • Changed document title to ECP5 and ECP5-5G sysDSP Usage Guide.
Using sysDSP	Updated Clarity Designer Flow section. Replaced Figure 3.3. Generating Distributed 18x18 Multiplier in Clarity Designer in Lattice Diamond Software
Technical Support Assistance	Updated Technical Support Assistance section.

Revision 1.0, March 2015

Section	Change Summary
All	Initial release.

www.latticesemi.com