ECP5 and ECP5-5G sysDSP Usage Guide

November 2015 Technical Note TN1267

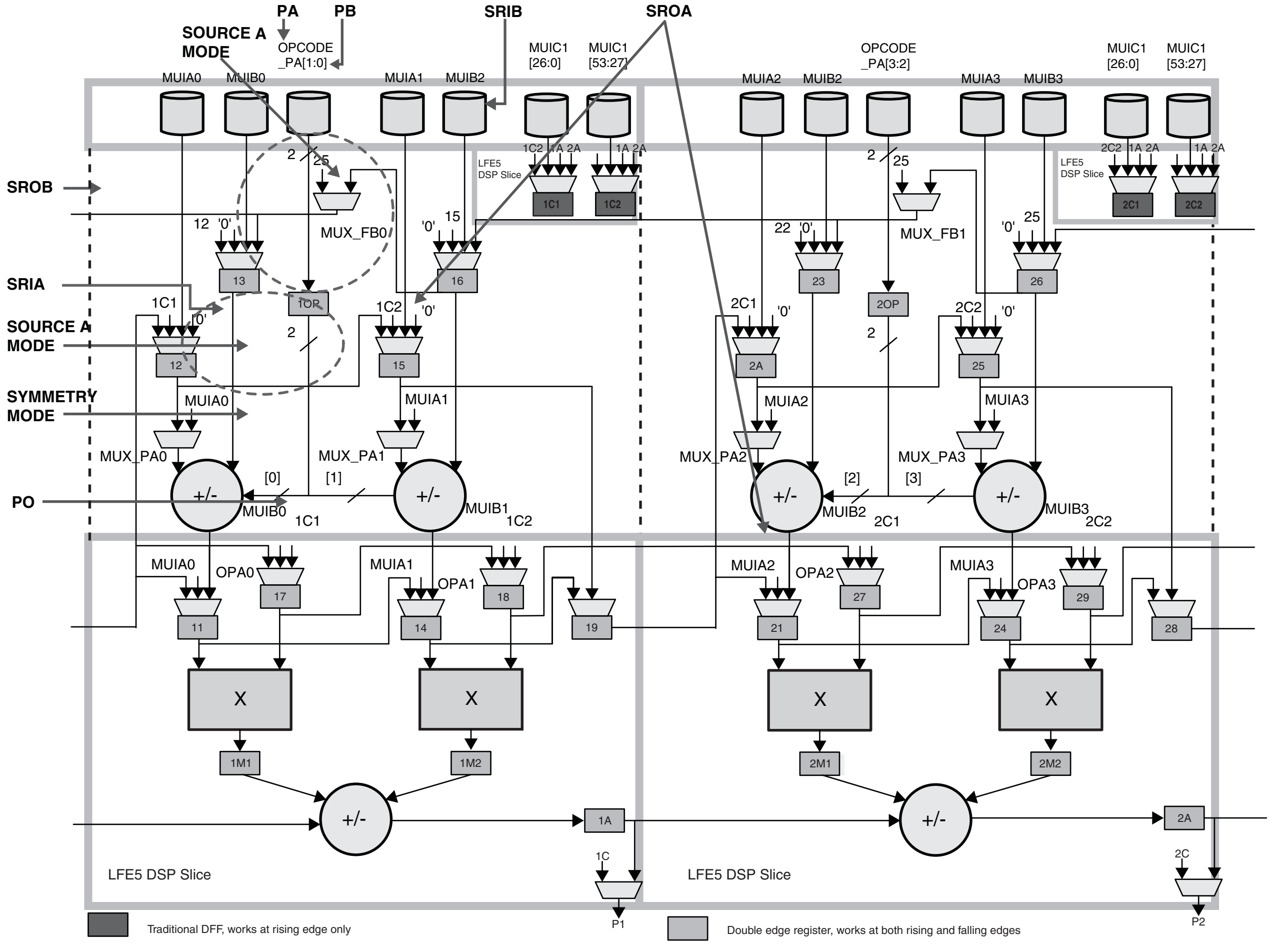
Introduction

This technical note discusses how to access the features of the ECP5™ and ECP5-5G™ sysDSP™ (Digital Signal Processing) slice described in DS1044, ECP5 and ECP5-5G Family Data Sheet. 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.

sysDSP Overview

Figure 1 shows the ECP5 and ECP5-5G device DSP Block Diagram 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, 54-bit ALU, output registers.

Figure 1. ECP5 and ECP5-5G DSP Block Diagram Overview



sysDSP slices are located in rows throughout the device. Figure 2 shows the simplified block diagram of the sysDSP slices. 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. ECP5 and ECP5-5G DSP Slice Detailed View

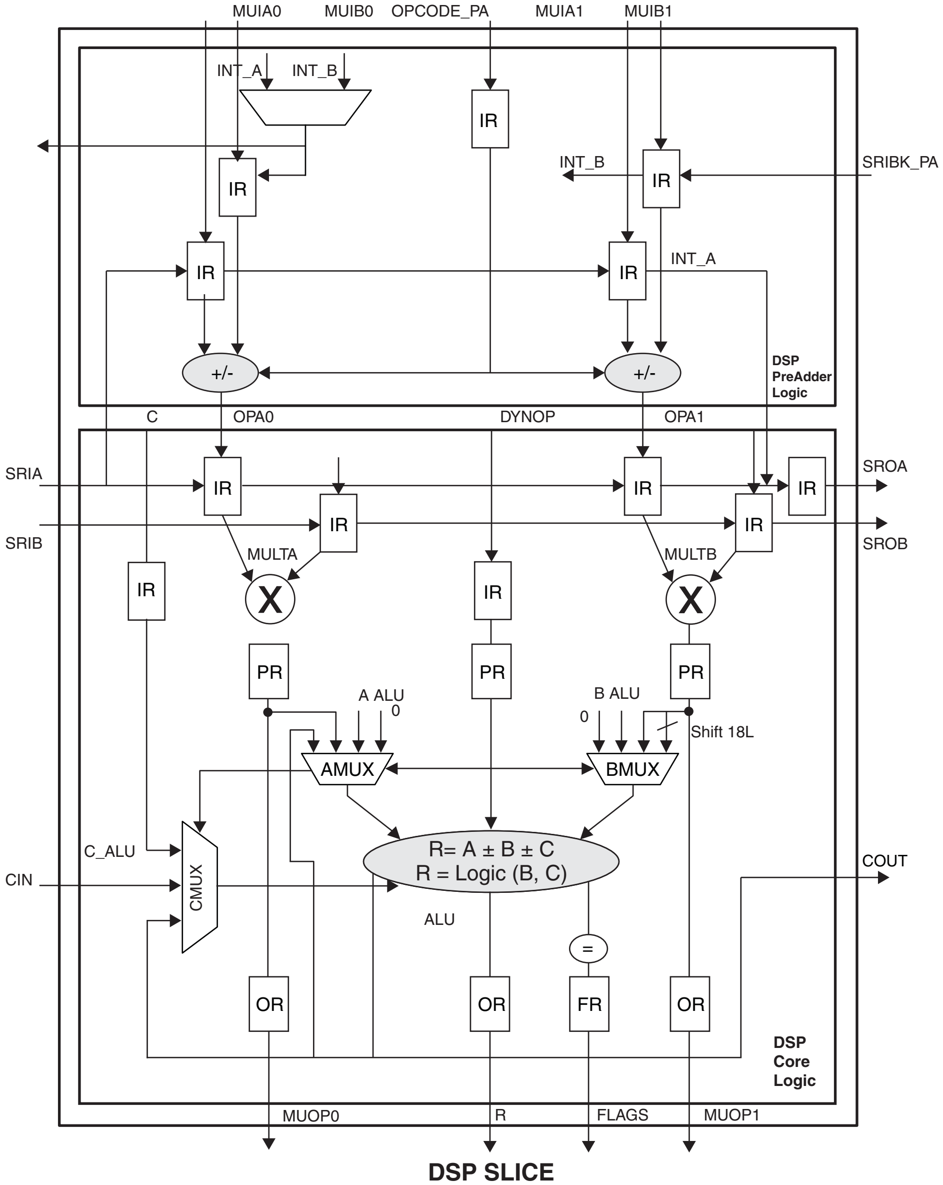


Figure 2 shows the individual ECP5 and ECP5-5G sysDSP slice in greater detail. It shows dual pre-adders with the core ECP5 and ECP5-5G DSP logic. The built-in pre-adders, multipliers and ALU minimize the amount of external logic required to implement some of the key DSP functions, resulting in efficient resource usage, reduced power consumption, improved performance, and data throughput for DSP applications. The ECP5 and ECP5-5G sysDSP slice can be configured several ways to suit users’ end applications.

The IR shown in a blue outline is an 18-bit register. The ORs and FR share a 72-bit register. If simple multiplier mode is implemented, the register is used as multiplier output. If ALU is implemented, it is used as ALU output.

Operating modes and features

The DSP Block has three main operating modes:

- One 36x36 Multiplier
  - Basic Multiplier, no add/sub/accumulator/sum blocks.
- Four 18x18 Multipliers
  - Two add/sub/accumulator blocks
  - One summation Block for adding four multipliers
- Eight 9x9 Multipliers
  - Four add/sub/accumulator blocks
  - Two Summation Blocks

Additionally, the device has advanced features such as:

- 18-bit dual multipliers
- 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
- Higher Operation of Frequency (400 MHz).

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 1.

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.

Primitive Instantiation sysDSP

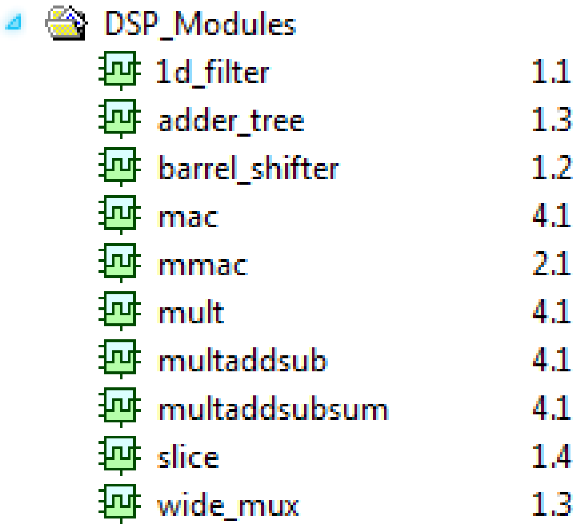
The sysDSP primitives can be directly instantiated in the design. Each of the primitives has a fixed set of attributes that can be customized to meet the design requirements.

An example of the primitive instantiation is given 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.

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. Here is a screenshot of the module selection for the memory modules under Clarity Designer in Lattice Diamond software.

Figure 3. DSP Modules in Clarity Designer



DSP_Modules

1d_filter	1.1
adder_tree	1.3
barrel_shifter	1.2
mac	4.1
mmac	2.1
mult	4.1
multaddsub	4.1
multaddsubsum	4.1
slice	1.4
wide_mux	1.3

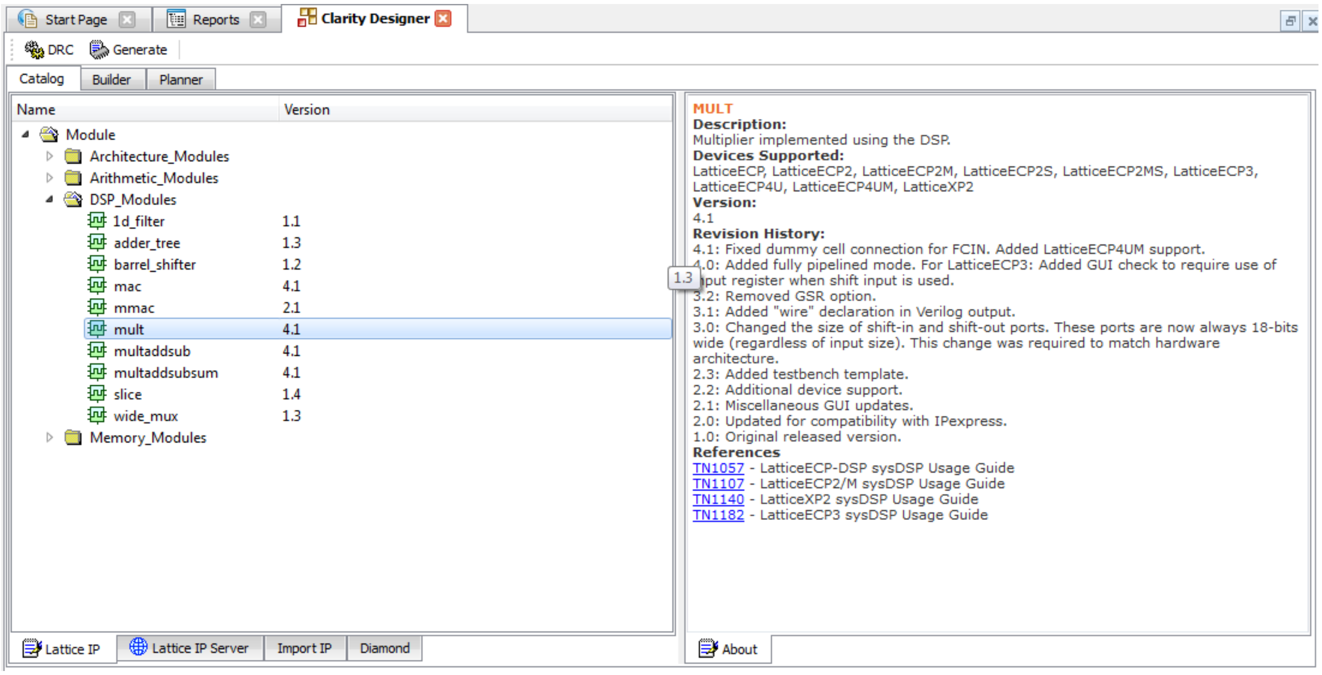
Clarity Designer Flow

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 4.

Figure 4. Clarity Designer in Lattice Diamond Software



Name	Version
Module	
Architecture_Modules	
Arithmetic_Modules	
DSP_Modules	
1d_filter	1.1
adder_tree	1.3
barrel_shifter	1.2
mac	4.1
mmac	2.1
mult	4.1
multaddsub	4.1
multaddsubsum	4.1
slice	1.4
wide_mux	1.3
Memory_Modules	

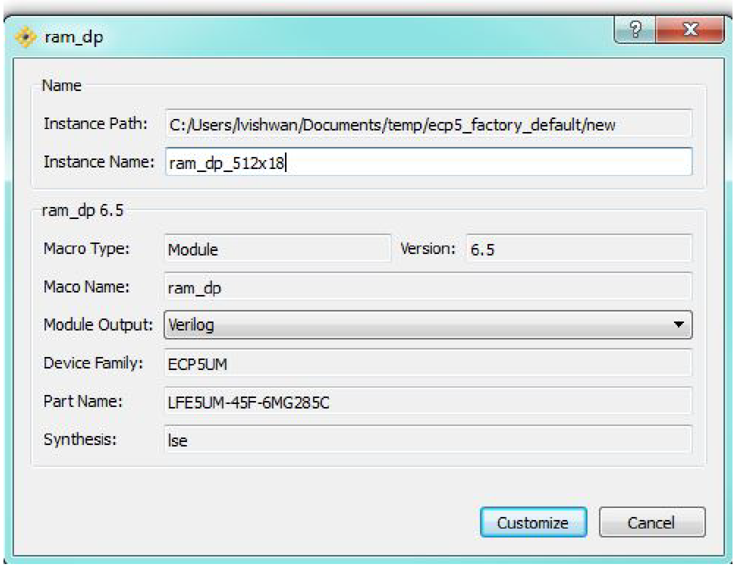
MULT
Description:
Multiplier implemented using the DSP.
Devices Supported:
LatticeECP, LatticeECP2, LatticeECP2M, LatticeECP2S, LatticeECP2MS, LatticeECP3, LatticeECP4U, LatticeECP4UM, LatticeXP2
Version:
4.1
Revision History:
4.1: Fixed dummy cell connection for FCIN. Added LatticeECP4UM support.
4.0: Added fully pipelined mode. For LatticeECP3: Added GUI check to require use of input register when shift input is used.
3.2: Removed GSR option.
3.1: Added "wire" declaration in Verilog output.
3.0: Changed the size of shift-in and shift-out ports. These ports are now always 18-bits wide (regardless of input size). This change was required to match hardware architecture.
2.3: Added testbench template.
2.2: Additional device support.
2.1: Miscellaneous GUI updates.
2.0: Updated for compatibility with IPexpress.
1.0: Original released version.
References
TN1057 - LatticeECP-DSP sysDSP Usage Guide
TN1107 - LatticeECP2/M sysDSP Usage Guide
TN1140 - LatticeXP2 sysDSP Usage Guide
TN1182 - LatticeECP3 sysDSP Usage Guide

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 18x18 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 5.

Figure 5. Generating Distributed 18x18 Multiplier in Clarity Designer in Lattice Diamond Software



ram_dp

Name

Instance Path: C:/Users/lvishwan/Documents/temp/ecp5_factory_default/new

Instance Name: ram_dp_512x18

ram_dp 6.5

Macro Type: Module

Version: 6.5

Maco Name: ram_dp

Module Output: Verilog

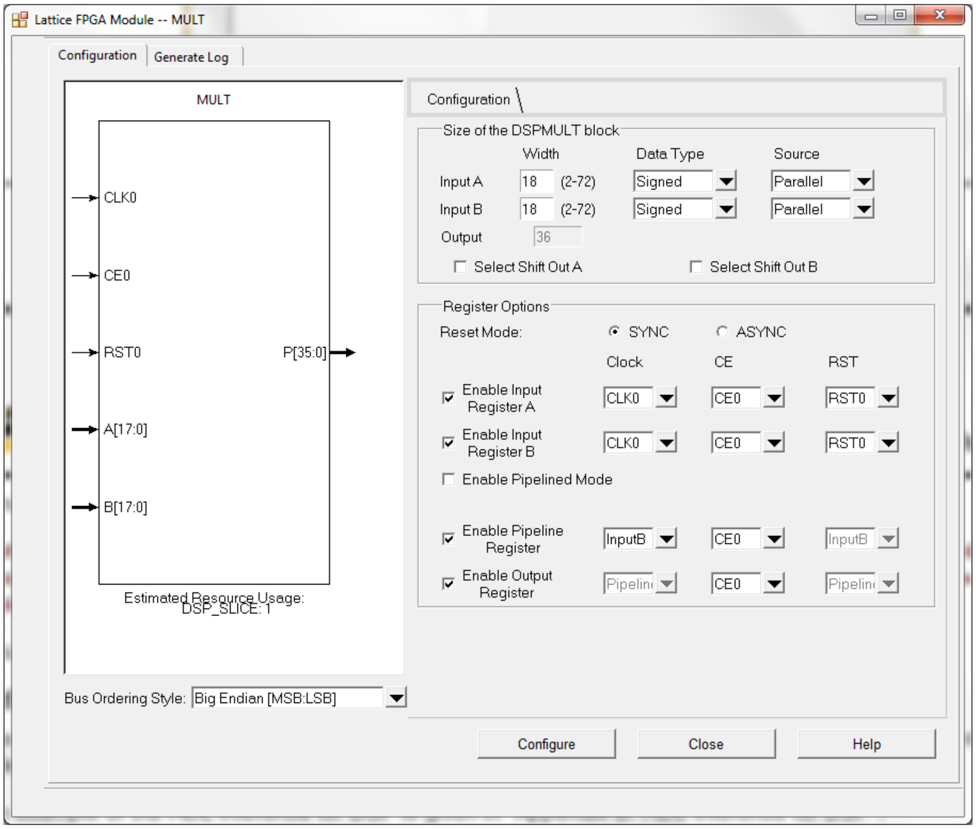
Device Family: ECP5UM

Part Name: LFE5UM-45F-6MG285C

Synthesis: lse

Click **Customize** to open another window, as shown in Figure 6, where you can customize the 18x18 Multiplier.

Figure 6. Customizing Multiplier in Clarity Designer in Lattice Diamond Software



Estimated Resource Usage:
DSP_SLICE: 1

Bus Ordering Style: Big Endian [MSB:LSB]

Configuration

Size of the DSPMULT block

	Width		Data Type	Source
Input A	18	(2-72)	Signed	Parallel
Input B	18	(2-72)	Signed	Parallel
Output	36			

□ Select Shift Out A □ Select Shift Out B

Register Options

Reset Mode: ● SYNC ○ ASYNC

	Clock	CE	RST
☑ Enable Input Register A	CLK0	CE0	RST0
☑ Enable Input Register B	CLK0	CE0	RST0
□ Enable Pipelined Mode			
☑ Enable Pipeline Register	InputB	CE0	InputB
☑ Enable Output Register	Pipelin	CE0	Pipelin

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.

Inferencing sysDSP slice

Designers can write a behavioral code for the DSP function such as multiplier, ALU etc., 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 given in Appendix B: HDL Inference for DSP.

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 extra coding and knowledge by the user. This section details the primitives supported by ECP5 and ECP5-5G devices. Please refer to Appendix A: Instantiating DSP Primitives in HDL that shows an HDL examples on how to instantiate sysDSP primitives.

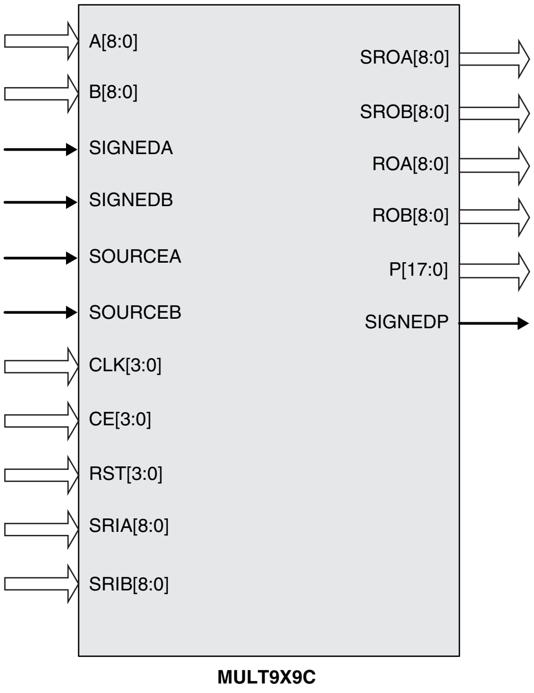
The ECP5 and ECP5-5G sysDSP supports all the legacy ECP5 and ECP5-5G device primitives, namely MULT9X9C, MULT18X18C, ALU24A and ALU24B. In addition, several other library primitives have been defined to take advantage of the features of the ECP5 and ECP5-5G sysDSP slice.

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

MULT9X9C – Advanced 9X9 DSP Multiplier

The 9x9 multiplier is a widely used module. Figure 7 shows the MULT9X9C primitive available in the ECP5 and ECP5-5G device.

Figure 7. MULT9X9C Primitive



MULT9X9C – I/O Port Description

Table 1 describes the list of ports available for MULT9X9C primitive.

Table 1. MULT9X9C I/O Port Description

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

MULT9X9C – Attribute Description

Table 2 describes the attributes for MULT9X9C primitive.

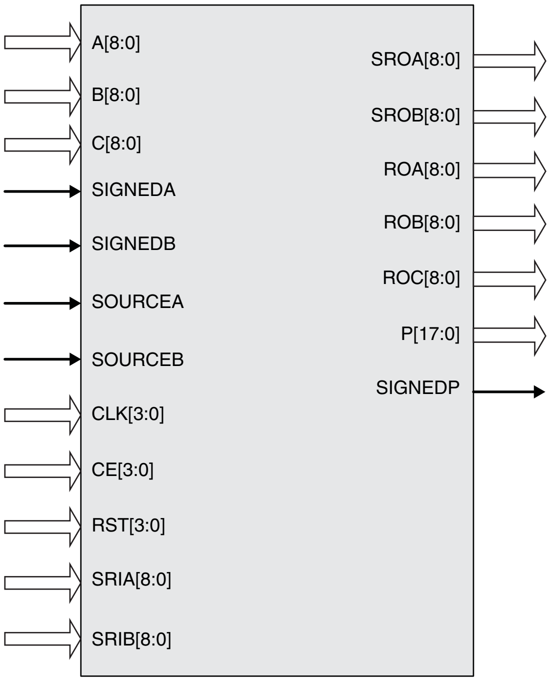
Table 2. Attribute Description for MULT9X9C

Attribute Name	Values	Default Value	GUI 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

MULT9X9D – Advanced 9X9 DSP Multiplier for Highspeed

This version of 9x9 multiplier has been optimized for high speed. Figure 8 shows the MULT9X9D primitive available in ECP5 and ECP5-5G device.

Figure 8. MULT9X9D Primitive



MULT9X9D

MULT9X9D – I/O Port Description

The Table 3 describes the list of ports available for MULT9X9D primitive.

Table 3. MULT9X9D I/O Port Description

Port	I/O	Description
A[8:0]	I	Multiplier parallel Input A
B[8:0]	I	Multiplier parallel Input B
C[8:0]	I	Multiplier Input C
SIGNEDA	I	Signed Bit for Input A
SIGNEDB	I	Signed Bit for Input B
SOURCEA	I	Source Selector for Multiplier Input A
SOURCEB	I	Source Selector for Multiplier Input B
CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	I	Clock Inputs
RST[3:0]	I	Reset Inputs
SRIA[8:0]	I	Multiplier shift Input A
SRIB[8:0]	I	Multiplier shift Input B
SROA[8:0]	O	Shift Output A
SROB[8:0]	O	Shift Output B
ROA[8:0]	O	Output A
ROB[8:0]	O	Output B
ROC[8:0]	O	Shift Output C – To be used for right side of the slice only
P[17:0]	O	Product Output
SIGNEDP	O	Signed Bit for the Product Output

MULT9X9D – Attribute Description

The Table 4 describes the attributes for MULT9X9D primitive.

Table 4. Attribute Description for MULT9X9D

Attribute Name	Values	Default Value	GUI 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_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_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
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, CLK0, CLK1, CLK2, CLK3	NONE	Y
GSR	ENABLED, DISABLED	ENABLED	N
CAS_MATCH_REG	TRUE, FALSE	FALSE	Y
SOURCEB_MODE	B_SHIFT, C_SHIFT, B_C_DYNAMIC, HIGHSPEED	B_SHIFT	Y
MULT_BYPASS	ENABLED, DISABLED	DISABLED	N
RESETMODE	SYNC, ASYNC	SYNC	Y

MULT9X9D has an option to select the source for the Multiplier Input B. Table 5 lists the details of SOURCEB_MODE Attribute for MULT18X18D Primitive

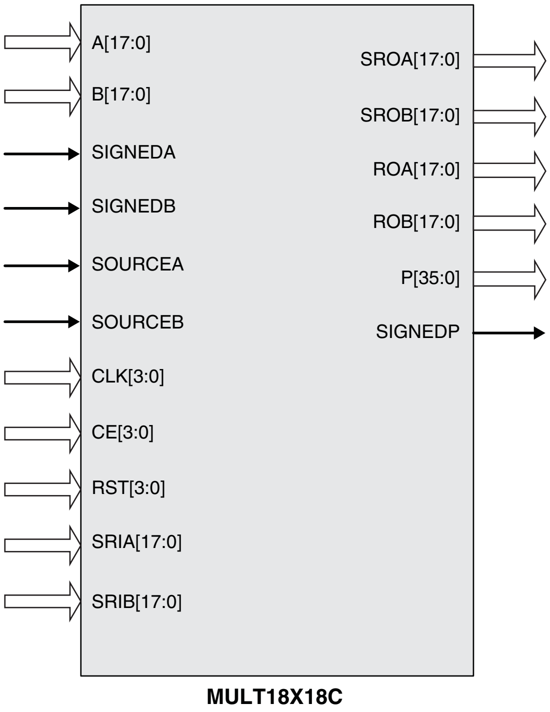
Table 5. SOURCEB_MODE Attribute for MULT18X18D Primitive

IP Express Operation	SOURCEB_MODE Attribute	SOURCEB Port	Mc1_b0_mux3	Mc1_0b_mux4
Shift	B_SHIFT	1	00	01
B	B_SHIFT	0	00	00
C	C_SHIFT	0	01	00
B/C Dynamic	B_C_DYNAMIC	Live	10	00
Highspeed BC	HIGHSPEED	0	11	00
Dynamic Shift/B	B_SHIFT	Live	00	10
Dynamic Shift/C	C_SHIFT	Live	01	10

MULT18X18C – Basic 18X18 DSP Multiplier

The ECP5 and ECP5-5G device also includes the 18X18 multiplier natively. Figure 9 shows the MULT18X18C primitive available in ECP5 and ECP5-5G device.

Figure 9. MULT18X18C Primitive



MULT18X18C – I/O Port Description

Table 6 describes the port list for MULT18X18C primitive.

Table 6. MULT18X18C I/O Port Description

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

MULT18X18C – Attribute Description

Table 7 describes the attributes for MULT18X18C primitive.

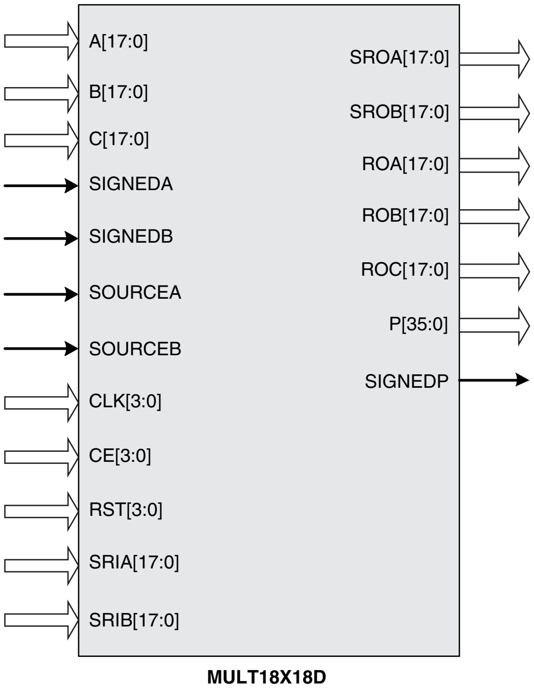
Table 7. Attribute Description for MULT18X18C

Attribute Name	Values	Default Value	GUI 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

MULT18X18D – Advanced 18X18 DSP Multiplier for High Speed

Similar to its 9X9 counterpart, 18X18 also has a high speed version – MULT18X18D. Figure 10 shows the MULT18X18D primitive

Figure 10. MULT18X18D Primitive



MULT18X18D – I/O Port Description

Table 8 describes the port list for MULT18X18D primitive.

Table 8. MULT18X18D I/O Port Description

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

MULT18X18D – Attribute Description

Table 9 describes the attributes for MULT18X18D primitive.

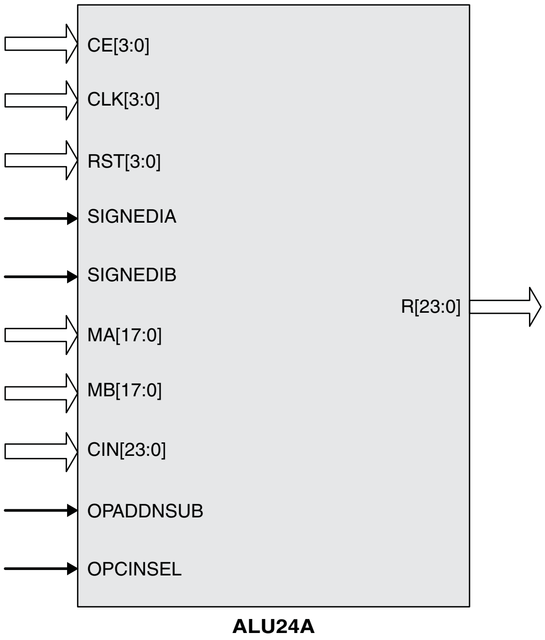
Table 9. Attribute Description for MULT18X18D

Attribute Name	Values	Default Value	GUI 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_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_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
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, CLK0, CLK1, CLK2, CLK3	NONE	Y
GSR	ENABLED, DISABLED	ENABLED	N
CAS_MATCH_REG	TRUE, FALSE	FALSE	Y
SOURCEB_MODE	B_SHIFT, C_SHIFT, B_C_DYNAMIC, HIGHSPEED	B_SHIFT	Y
MULT_BYPASS	ENABLED, DISABLED	DISABLED	N
RESETMODE	SYNC, ASYNC	SYNC	Y

ALU24A – 24-bit Ternary Adder/ Subtractor

ECP5 and ECP5-5G devices also allows configuration in an ALU mode. Figure 11 shows the ALU24A primitive

Figure 11. ALU24A Primitive



ALU24A – I/O Port Description

Table 10 describes the port list for ALU24A primitive.

Table 10. ALU24A I/O Port Description

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

ALU24A – Attribute Description

Table 11 describes the attributes for ALU24A primitive.

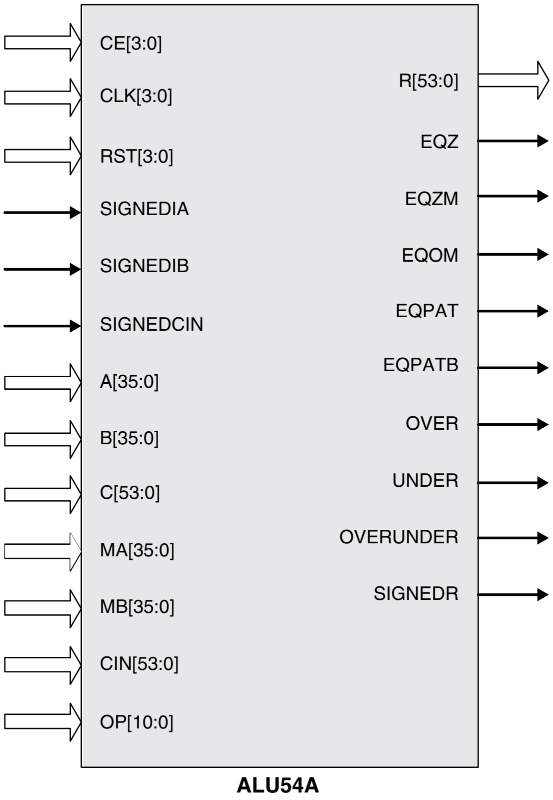
Table 11. Attribute Description for ALU24A

Attribute Name	Values	Default Value	GUI 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

ALU54A – 54-bit Ternary Adder/ Subtractor

Figure 12 shows the ALU54A primitive

Figure 12. ALU54APrimitive



ALU24A – I/O Port Description

Table 12 describes the port list for ALU54A primitive.

Table 12. ALU54A I/O Port Description

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

ALU54A – Attribute Description

Table 13 describes the attributes for ALU54A primitive.

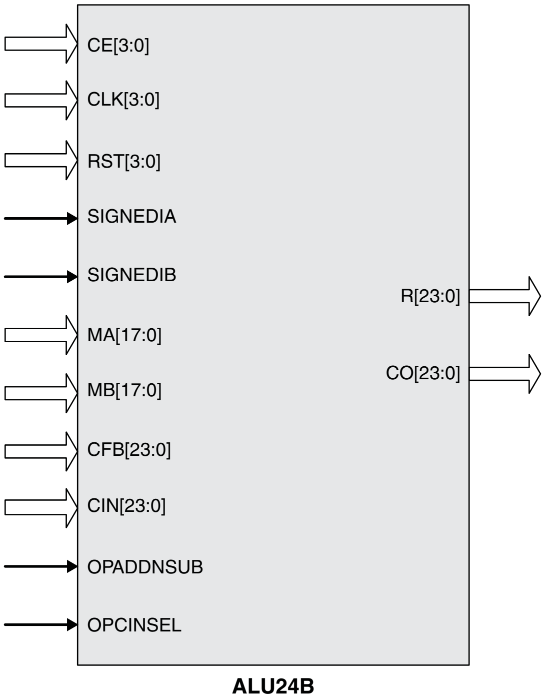
Table 13. Attribute Description for ALU54A

Attribute Name	Values	Default Value	GUI 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 0xFFFFFFFFFFFFFF	0x00000000000000	Y
MCPAT	0x00000000000000 to 0xFFFFFFFFFFFFFF	0x00000000000000	Y
MASKPAT	0x00000000000000 to 0xFFFFFFFFFFFFFF	0x00000000000000	Y
RNDPAT	0x00000000000000 to 0xFFFFFFFFFFFFFF	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

ALU24B – 24-bit Ternary Adder/ Subtractor for 9X9 Mode

Figure 13 shows the ALU24B primitive.

Figure 13. ALU24B Primitive



ALU24B – I/O Port Description

Table 14 describes the port list for ALU24B primitive.

Table 14. ALU24B I/O Port Description

Port	I/O	Description
CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	I	Clock Inputs
RST[3:0]	I	Reset Inputs
SIGNEDIA	I	Sign Bit for Input A
SIGNEDIB	I	Sign Bit for Input B
MA[17:0]	I	Input A
MB[17:0]	I	Input B
CFB[23:0]	I	C Input for Highspeed
CIN[23:0]	I	Carry In Input
OPADDNSUB	I	Add/Sub Selector
OPCINSEL	I	CarryIn Selector
R[23:0]	O	Sum
CO[23:0]	O	Sum – Special Routing output used for Highspeed option

ALU24B – Attribute Description

Table 15 describes the attributes for ALU24B primitive.

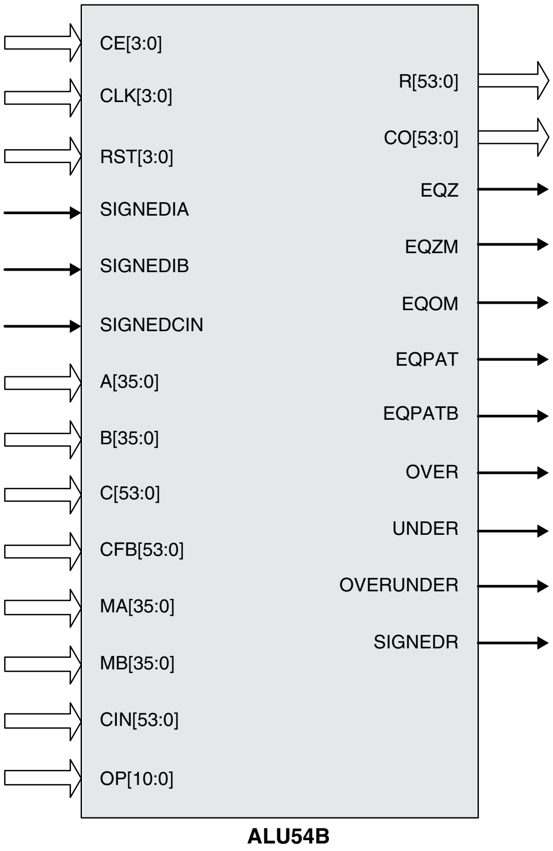
Table 15. Attribute Description for ALU24B

Attribute Name	Values	Default Value	GUI 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
REG_INPUTCFB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTCFB_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTCFB_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
RESETMODE	SYNC, ASYNC	SYNC	Y
GSR	ENABLED, DISABLED	ENABLED	N

ALU54B – 54-bit Ternary Adder/ Subtractor for High Speed

Figure 14 shows the ALU54B primitive

Figure 14. ALU54B Primitive



ALU54B – I/O Port Description

Table 16 describes the port list for ALU54B primitive.

Table 16. ALU54B I/O Port Description

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

ALU54B – Attribute Description

Table 17 describes the attributes for ALU54B primitive.

Table 17. Attribute Description for ALU54B

Attribute Name	Values	Default Value	GUI 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_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 0xFFFFFFFFFFFFFF	0x00000000000000	Y
REG_INPUTCFB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTCFB_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTCFB_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
MCPAT	0x00000000000000 to 0xFFFFFFFFFFFFFF	0x00000000000000	Y

Attribute Name	Values	Default Value	GUI Access
MASKPAT	0x00000000000000 to 0xFFFFFFFFFFFFFF	0x00000000000000	Y
RNDPAT	0x00000000000000 o 0xFFFFFFFFFFFFFF	0x00000000000000	Y
GSR	ENABLED, DISABLED	ENABLED	N
RESETMODE	SYNC, ASYNC	SYNC	Y
MULT9_MODE	ENABLED, DISABLED	DISABLED	N
FORCE_ZERO_BARREL_SHIFT	ENABLED, DISABLED	DISABLED	N
LEGACY	ENABLED, DISABLED	DISABLED	Y

In case of ALU54B, it has to be noted that the REG_INPUT_C0 corresponds to the lower 27 bits of the C Input, REG_INPUT_C1 corresponds to the upper 27 bits of the C Input, and REG_OUTPUT0* corresponds to [17:0] of R and REG_OUTPUT1_* corresponds to [53:18] of R.

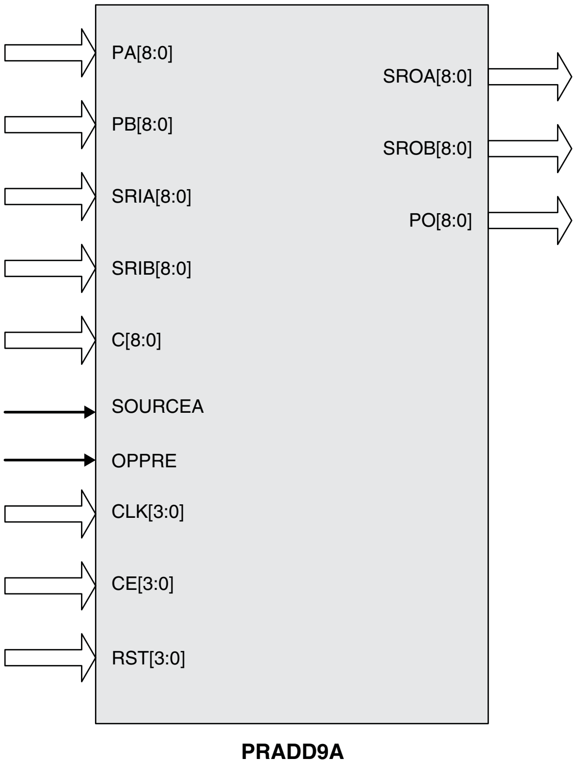
Also, when REG_INPUTCFB_CLK = NONE, it means that the CFB ports are not used, and C -> Cr uses the C0/C1_CLK attributes.

When REG_INPUTCFB_CLK != NONE, the CFB ports are being used, CFB -> CO is using these attributes and C -> Cr is unregistered.

PRADD9A – 9-bit Pre-Adder/Shift

Figure 15 shows the PRADD9A primitive.

Figure 15. PRADD9A Primitive



PRADD9A – I/O Port Description

Table 18 describes the port list for PRADD9A primitive.

Table 18. PRADD9A I/O Port Description

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

PRADD9A – Attribute Description

Table 19 describes the attributes for PRADD9A primitive.

Table 19. Attribute Description for PRADD9A

Attribute Name	Values	Default Value	GUI 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, CLK0, CLK1, CLK2, CLK3	NONE	Y	
GSR	ENABLED, DISABLED	ENABLED	N	
CAS_MATCH_REG	TRUE, FALSE	FALSE	Y	
SOURCEA_MODE	A_SHIFT, C_SHIFT, A_C_DYNAMIC, HIGHSPEED	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 20. The other Source mode attributes and the Feedback Mux information are also included in Table 20.

Table 20. SOURCEB_MODE Attribute for PRADD9A Primitive

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
HighspeedAC	HIGHSPEED	0	11	00
Dynamic Shift/A	A_SHIFT	Live	00	10
Dynamic Shift/C	C_SHIFT	Live	01	10

Table 21. Details of SOURCEB_MODE Attribute

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 22. Details of FB_MUX Attribute

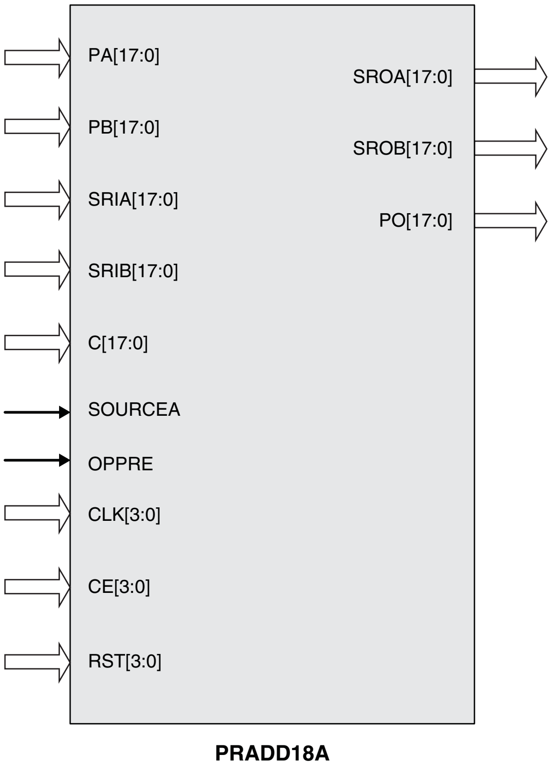
FB_MUX Attribute	Operation (MUX_FB0)
SHIFT	Output of Reg. 16
SHIFT_BYPASS	Output of Reg. 15
DISABLED	For placer only (PreAdder on the left side)

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.

PRADD18A – 18-bit Pre-Adder/Shift

Figure 16 shows the PRADD18A primitive

Figure 16. PRADD18A Primitive



PRADD9A – I/O Port Description

The Table 23 describes the port list for PRADD18A primitive.

Table 23. PRADD18A I/O Port Description

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

PRADD9A – Attribute Description

The Table 24 describes the attributes for PRADD18A primitive.

Table 24. Attribute Description for PRADD18A

Attribute Name	Values	Default Value	GUI 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, CLK0, CLK1, CLK2, CLK3	NONE	Y	
GSR	ENABLED, DISABLED	ENABLED	N	
CAS_MATCH_REG	TRUE, FALSE	FALSE	Y	
SOURCEA_MODE	A_SHIFT, C_SHIFT, A_C_DYNAMIC, HIGHSPEED	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	
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 the Table 25. The other Source mode attributes and the Feedback Mux information is also includes the tables that follow.

Table 25. Details of SOURCEA_MODE Attribute

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
HighspeedAC	HIGHSPEED	0	11	00
Dynamic Shift/A	A_SHIFT	Live	00	10
Dynamic Shift/C	C_SHIFT	Live	01	10

Table 26. Details of SOURCEB_MODE Attribute

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 27. Details of FB_MUX Attribute

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.

Technical Support Assistance

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

Revision History

Date	Version	Change Summary
November 2015	1.1	Added support for ECP5-5G.
		Changed document title to ECP5 and ECP5-5G sysDSP Usage Guide.
		Updated Clarity Designer Flow section. Replaced Figure 5, Generating Distributed 18x18 Multiplier in Clarity Designer in Lattice Diamond Software.
		Updated Technical Support Assistance section.
March 2014	01.0	Initial release.

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 MULT18X18D Instantiation

```
defparam dsp_mult_0.CLK3_DIV = "DISABLED" ;
defparam dsp_mult_0.CLK2_DIV = "DISABLED" ;
defparam dsp_mult_0.CLK1_DIV = "DISABLED" ;
defparam dsp_mult_0.CLK0_DIV = "DISABLED" ;
defparam dsp_mult_0.HIGHSPEED_CLK = "CLK0" ;
defparam dsp_mult_0.REG_INPUTC_RST = "RST0" ;
defparam dsp_mult_0.REG_INPUTC_CE = "CE0" ;
defparam dsp_mult_0.REG_INPUTC_CLK = "NONE" ;
defparam dsp_mult_0.SOURCEB_MODE = "B_SHIFT" ;
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" ;
MULT18X18D 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), .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), .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), .ROC17(), .ROC16(), .ROC15(),
   .ROC14(), .ROC13(), .ROC12(), .ROC11(), .ROC10(), .ROC9(), .ROC8(),
   .ROC7(), .ROC6(), .ROC5(), .ROC4(), .ROC3(), .ROC2(), .ROC1(), .ROC0(),
   .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 ALU54B Instantiation

```
dsp_alu_0: ALU54B
   generic map (
      CLK3_DIV=> "DISABLED", CLK2_DIV=> "DISABLED",
      CLK1_DIV=> "DISABLED", CLK0_DIV=> "DISABLED", REG_INPUTCFB_RST=> "RST0",
      REG_INPUTCFB_CE=> "CE0", REG_INPUTCFB_CLK=> "CLK1",
      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,
      CFB53=>r5_53, CFB52=>r5_52, CFB51=>r5_51, CFB50=>r5_50,
      CFB49=>r5_49, CFB48=>r5_48, CFB47=>r5_47, CFB46=>r5_46,
      CFB45=>r5_45, CFB44=>r5_44, CFB43=>r5_43, CFB42=>r5_42,
      CFB41=>r5_41, CFB40=>r5_40, CFB39=>r5_39, CFB38=>r5_38,
      CFB37=>r5_37, CFB36=>r5_36, CFB35=>r5_35, CFB34=>r5_34,
      CFB33=>r5_33, CFB32=>r5_32, CFB31=>r5_31, CFB30=>r5_30,
      CFB29=>r5_29, CFB28=>r5_28, CFB27=>r5_27, CFB26=>r5_26,
CFB25=>r5_25, CFB24=>r5_24, CFB23=>r5_23, CFB22=>r5_22,
CFB21=>r5_21, CFB20=>r5_20, CFB19=>r5_19, CFB18=>r5_18,
CFB17=>r5_17, CFB16=>r5_16, CFB15=>r5_15, CFB14=>r5_14,
CFB13=>r5_13, CFB12=>r5_12, CFB11=>r5_11, CFB10=>r5_10,
CFB9=>r5_9, CFB8=>r5_8, CFB7=>r5_7, CFB6=>r5_6, CFB5=>r5_5,
CFB4=>r5_4, CFB3=>r5_3, CFB2=>r5_2, CFB1=>r5_1, CFB0=>r5_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, CO53=>Result1(53), CO52=>Result1(52),
CO51=>Result1(51), CO50=>Result1(50), CO49=>Result1(49),
CO48=>Result1(48), CO47=>Result1(47), CO46=>Result1(46),
CO45=>Result1(45), CO44=>Result1(44), CO43=>Result1(43),
CO42=>Result1(42), CO41=>Result1(41), CO40=>Result1(40),
CO39=>Result1(39), CO38=>Result1(38), CO37=>Result1(37),
CO36=>Result1(36), CO35=>Result1(35), CO34=>Result1(34),
CO33=>Result1(33), CO32=>Result1(32), CO31=>Result1(31),
CO30=>Result1(30), CO29=>Result1(29), CO28=>Result1(28),
CO27=>Result1(27), CO26=>Result1(26), CO25=>Result1(25),
CO24=>Result1(24), CO23=>Result1(23), CO22=>Result1(22),
CO21=>Result1(21), CO20=>Result1(20), CO19=>Result1(19),
CO18=>Result1(18), CO17=>Result1(17), CO16=>Result1(16),
CO15=>Result1(15), CO14=>Result1(14), CO13=>Result1(13),
CO12=>Result1(12), CO11=>Result1(11), CO10=>Result1(10),
CO9=>Result1(9), CO8=>Result1(8), CO7=>Result1(7),
CO6=>Result1(6), CO5=>Result1(5), CO4=>Result1(4),
CO3=>Result1(3), CO2=>Result1(2), CO1=>Result1(1),
CO0=>Result1(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 will not be optimal. Users 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
```