LatticeECP3 sysDSP Usage Guide

Technical Note

FPGA-TN-02193-1.4

March 2024

Disclaimers

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Abbreviation	Definition
ALU	Arithmetic Logic Unit
DSP	Digital Signal Processing
FIR	Finite Impulse Response
GUI	Graphical User Interface
HDL	Hardware Description Language
LUT	Look Up Table
TDM	Time Division Multiplexing

1. Introduction

This technical note discusses how to access the features of the LatticeECP3™ sysDSP™ (Digital Signal Processing) slice described in the LatticeECP3 Family Data Sheet (FPGA-DS-02074). Designs targeting the sysDSP slice can offer significant improvement over traditional LUT-based implementations. Table 1.1 provides an example of the performance and area benefits of this approach.

Table 1.1. sysDSP Slice vs. LUT-based Multipliers

Multiplier Width	Register Pipelining	LatticeECP3-95-8 Using sysDSP Slice(s)		LatticeECP3-95-8 Using LUTs	
		f_{MAX} (MHz)	LUTs	f_{MAX} (MHz)	LUTs
9x9	Input, Multiplier, Output	402	0	268	114
18x18	Input, Multiplier, Output	402	0	163	411
36x18	Input, Multiplier, Output	394	0	123	737
36x36	Input, Multiplier, Output	225	0	105	1502

2. sysDSP Slice Hardware

The LatticeECP3 sysDSP slices are located in rows throughout the device. Figure 2.1 is a simplified block diagram of two of the sysDSP slices. The programmable resources in a slice include: multipliers, ALU, muxes, pipeline registers, shift register chain and cascade chain. If the shift out register A is selected, the cascade match register (Casc A0) is available. The multipliers can be configured as 18X18 or 9X9 and the ALU can be configured as 54-bit or 24-bit. Advanced features of the sysDSP slice are described later in this document.

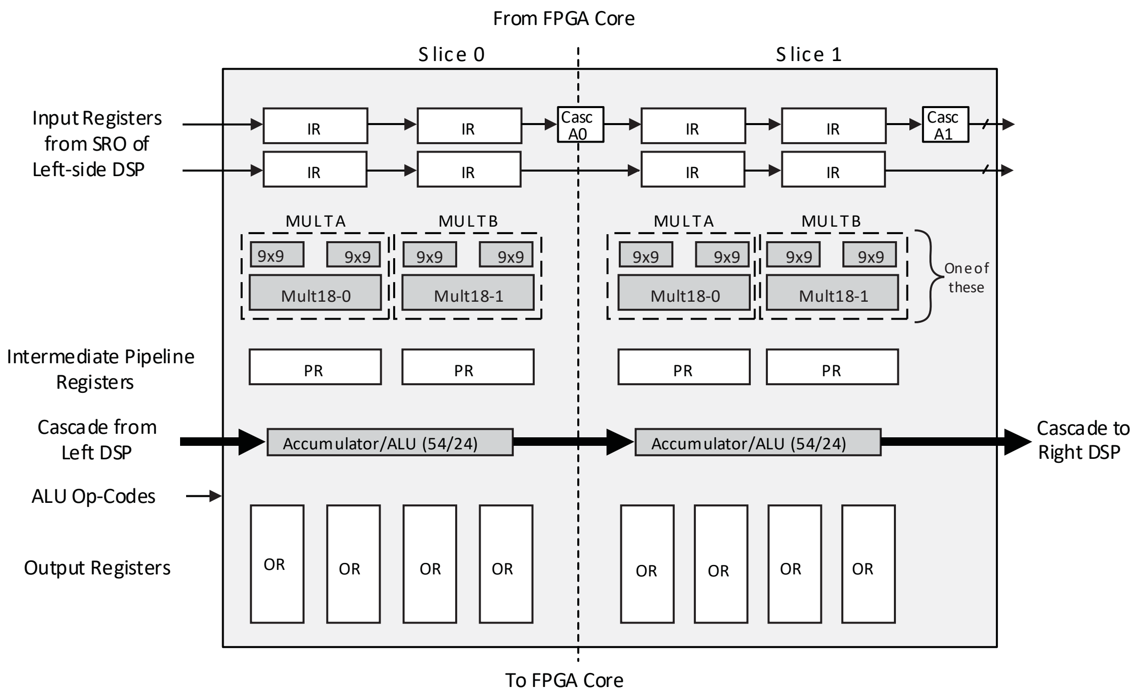


Figure 2.1. LatticeECP3 sysDSP Slice

The sysDSP slice can be configured in a number of ways. The ALU54/24 can be configured as one of the following: adder, subtracter, or accumulator. Using two sysDSP slices, the most common configurations include:

- One 36x36 Multiplier
  - Basic multiplier, no add/sub/accum/sum blocks
- Four 18x18 Multipliers
  - Two add/sub/accum blocks
- Eight 9x9 Multipliers
  - Two add/sub blocks

3. sysDSP Slice Software

3.1. Overview

The sysDSP slice can be targeted in a number of ways.

- Use IPexpress™ to specify and configure the sysDSP module for instantiating in the user HDL design.
- Create HDL code for direct sysDSP slice inference by the synthesis tools.
- Implement the design in The MathWorks® Simulink® tool using a Lattice library of DSP blocks. The ispLeverDSP™ tool in the ispLEVER® and Lattice Diamond ™ design software converts these blocks into HDL.
- Instantiate sysDSP primitives directly in the source code.

3.2. Targeting sysDSP Slices Using IPexpress

IPexpress allows you to graphically specify sysDSP elements. Once the element is specified, an HDL file is generated, which can be instantiated in a design. IPexpress allows users to configure all ports and set all available parameters. The following modules target the sysDSP slice. The resource usage estimation will be shown in the GUI. See Appendix B. Using IPexpress for Diamond for information about using IPexpress for Diamond.

- ADDER_TREE
- BARREL_SHIFTER
- MAC (Multiplier Accumulate)
- MMAC (Multiplier Multiplier Accumulate)
- MULT (Multiplier)
- MULTADDSUB (Multiplier Add/Subtract)
- MULTADDSUBSUM (Multiply Add/Subtract and SUM)
- SLICE (Fully-configurable sysDSP slice used for advanced functions)
- WIDE_MUX

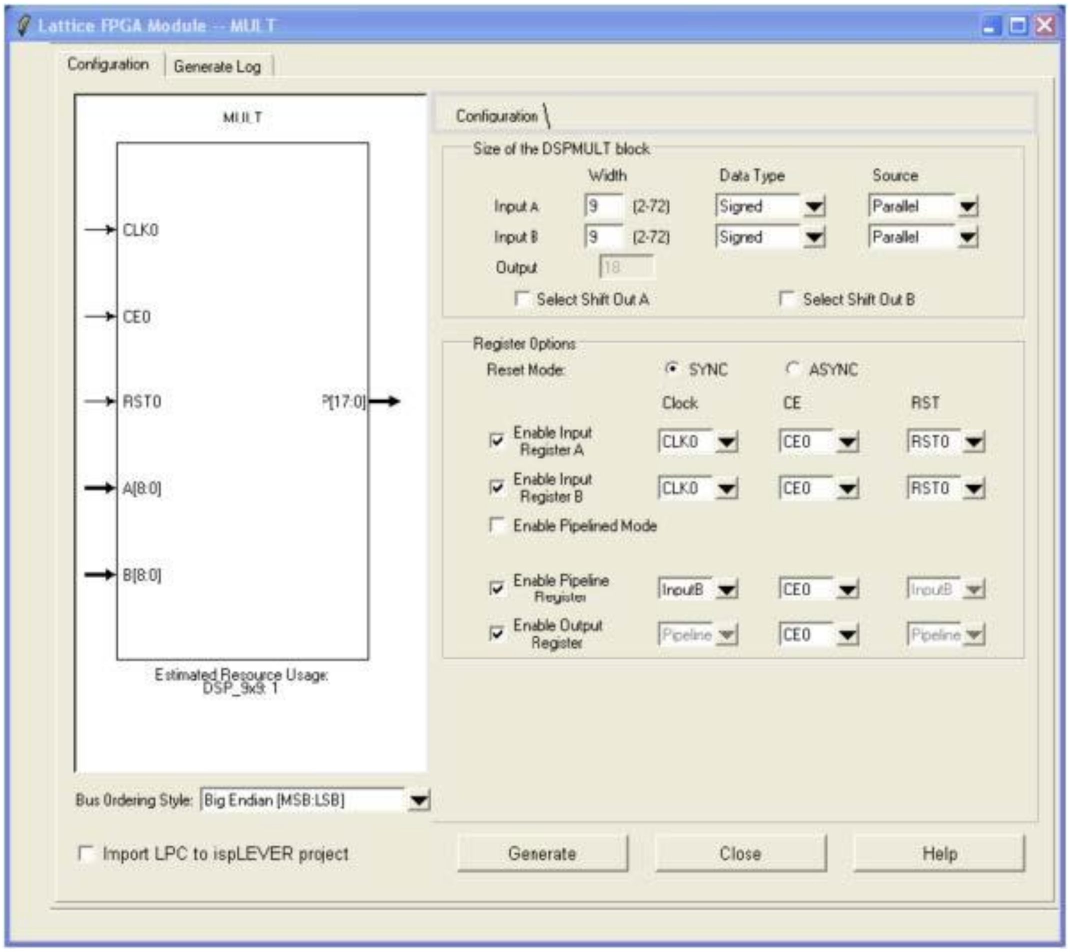
MULT Module

The IPexpress MULT Module configures elements to be packed into the sysDSP slice. The function A x B = P is implemented. The screen shot shown in Figure 3.1 shows the following:

- **Data Options**
  - Input bus widths from 2 to 72 bits
  - Data Type specifies whether the data is Signed, Unsigned or Dynamic
  - Source specifies whether data is from Parallel, Shift or Dynamic
  - Shift Out enables the shift out port on the sysDSP slice
- **Register Options**
  - Reset Mode selects whether an Async or Sync Reset is used.
  - Enable Input Register selects whether data input registers are used. If enabled, clock, reset and clock enable resources can be selected.
  - Enable Pipeline Register selects whether data pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected. If the option InputA or InputB is selected, the pipeline register uses the same clock as the input register. Output registers will automatically be enabled and will use the same clock.

- Enable Output Register selects whether the data output pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected.
- Enable Pipelined Mode turns on all the input, pipeline and output registers. If this mode is enabled, latency will be provided.

Multiple sysDSP slices can be spanned to accommodate large functions. Additional LUTs may be required if multiple sysDSP slices are needed. The input data format can be selected as Parallel, Shift or Dynamic. The Shift format can only be enabled if inputs are less than 18 bits. The Shift format enables a sample/shift register, which is useful in applications such as the FIR Filter.



Estimated Resource Usage:
DSP_9x9: 1

Size of the DSPMULT block

	Width		Data Type	Source
Input A	9	(2-72)	Signed	Parallel
Input B	9	(2-72)	Signed	Parallel
Output	18			

☐ 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	Pipeline	CE0	Pipeline

Bus Ordering Style: Big Endian [MSB:LSB]

☐ Import LPC to ispLEVER project

Figure 3.1. MULT Mode

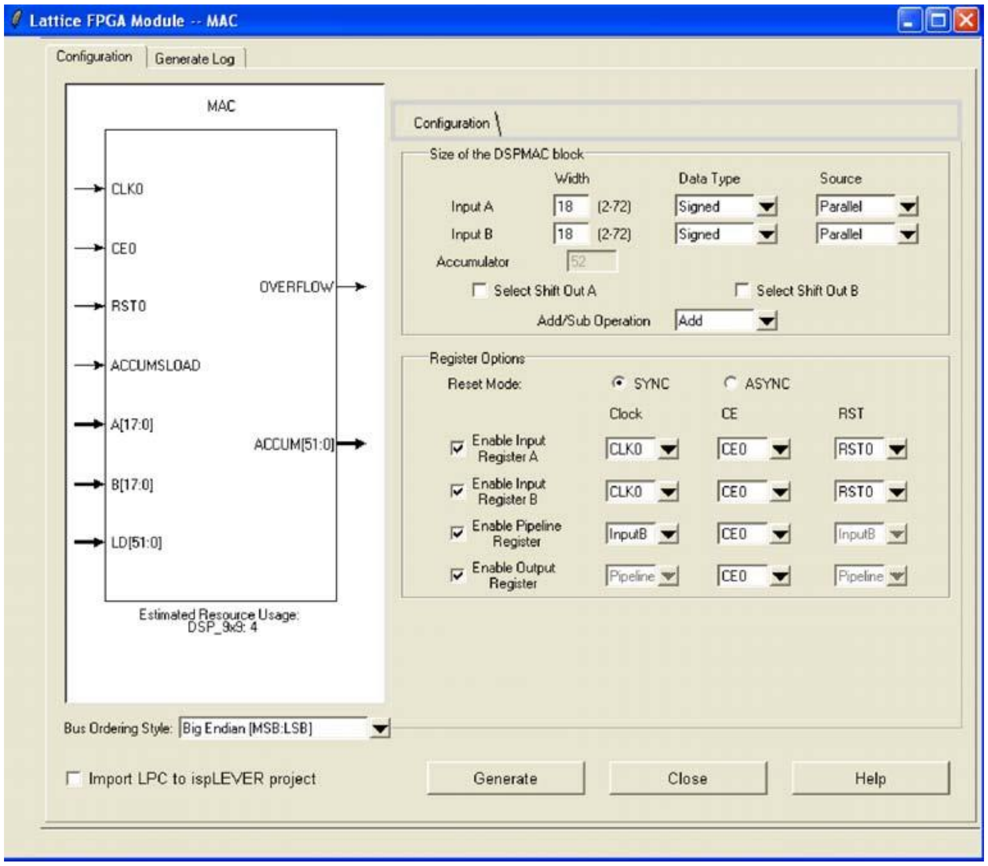
MAC Module

The IPexpress MAC Module configures elements to be packed into the sysDSP slice. The function An x Bn +/- Pn-1 = Pn is implemented. The screen shot shown in Figure 3.2 shows the following:

- **Data Options**
  - Input bus widths from 2 to 72 bits
  - Data Type specifies whether the data is Signed, Unsigned or Dynamic
  - Source specifies whether data is from Parallel, Shift or Dynamic
  - Shift Out enables the shift out port on the sysDSP slice
  - Add/Sub Operation selects whether the arithmetic operation is Addition, Subtraction or Dynamic
- **Register Options**
  - Reset Mode selects whether an Async or Sync Reset is used
  - Enable Input Register selects whether data input registers are used. If enabled, clock, reset and clock enable resources can be selected.

- Enable Pipeline Register selects whether data pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected. If the option InputA or InputB is selected, the pipeline register uses the same clock as the input register. Output registers will automatically be enabled and will use the same clock.
- Enable Output Register is selected by default and selects the data output pipeline registers to be used. Clock, reset and clock enable resources can be selected.

Multiple sysDSP slices can be spanned to accommodate large functions. Additional LUTs may be required if multiple sysDSP slices are needed. The input data format can be selected as Parallel, Shift or Dynamic. The Shift format can only be enabled if inputs are less than 18 bits. The Shift format enables a sample/shift register. The accumulator is loaded with an initial value from data on the LD port when the signal ACCUMSLOAD is toggled.



Size of the DSPMAC block

	Width		Data Type	Source
Input A	18	(2-72)	Signed	Parallel
Input B	18	(2-72)	Signed	Parallel
Accumulator	52			

☐ Select Shift Out A ☐ Select Shift Out B

Add/Sub Operation: Add

Register Options

Reset Mode: [✓ SYNC] ASYNC

	Clock	CE	RST
☑ Enable Input Register A	CLK0	CE0	RST0
☑ Enable Input Register B	CLK0	CE0	RST0
☑ Enable Pipeline Register	InputB	CE0	InputB
☑ Enable Output Register	Pipeline	CE0	Pipeline

Bus Ordering Style: Big Endian [MSB:LSB]

☐ Import LPC to ispLEVER project

Figure 3.2. MAC Mode

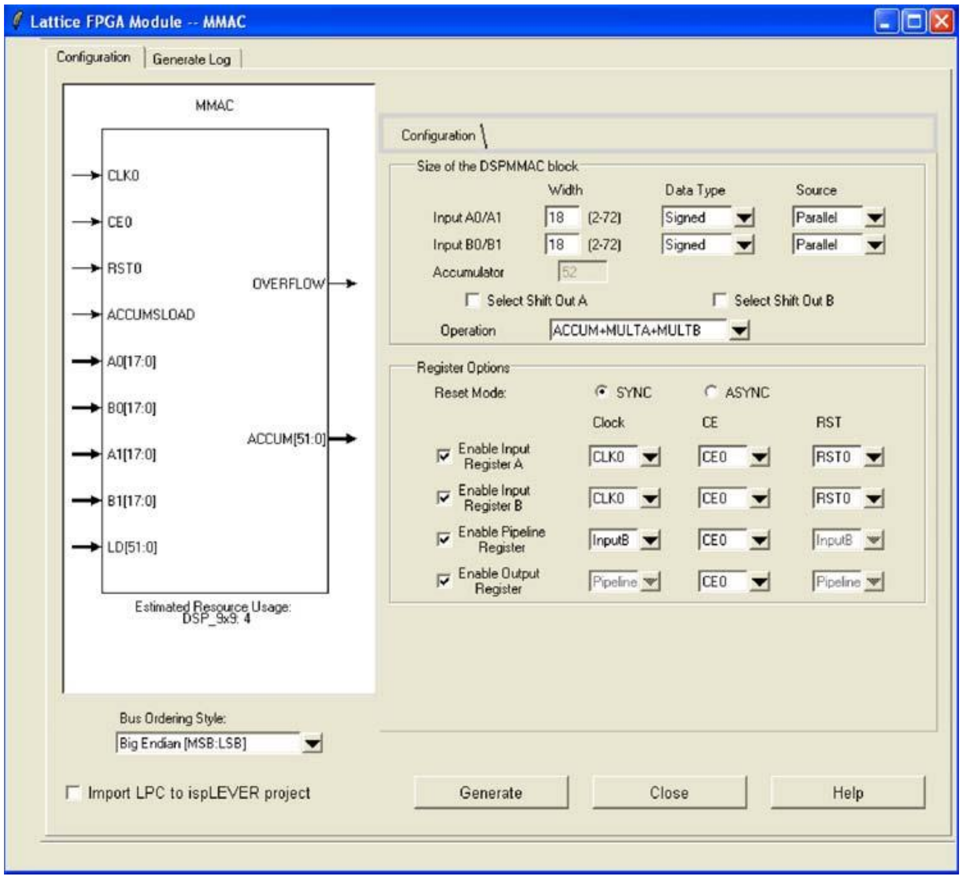
MMAC Module

The IPexpress MMAC Module configures elements to be packed into the sysDSP slice. The function A0n x B0n +/- A1n x B1n + Pn-1 = Pn is implemented. The screen shot shown in Figure 3.3 shows the following:

- **Data Options**
  - Input bus widths from 2 to 72 bits
  - Data Type specifies whether the data is Signed, Unsigned or Dynamic
  - Source specifies whether data is from Parallel, Shift (DSP slice shift out port) or Dynamic
  - Shift Out enables the shift out port on the DSP slice
  - Operation selects whether the arithmetic operation is Addition, Subtraction or Dynamic
- **Register Options**
  - Reset Mode selects whether an Async or Sync Reset is used.
  - Enable Input Register selects whether data input registers are used. If enabled, clock, reset and clock enable resources can be selected.

- Enable Pipeline Register selects whether data pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected. If the option InputA or InputB is selected, the pipeline register uses the same clock as the input register. Output registers will automatically be enabled and will use the same clock.
- Enable Output Register is selected by default and selects the data output pipeline registers to be used. Clock, reset and clock enable resources can be selected.

Multiple sysDSP slices can be spanned to accommodate large functions. Additional LUTs may be required if multiple sysDSP slices are needed. The input data format can be selected as Parallel, Shift or Dynamic. The Shift format can only be enabled if inputs are less than 18 bits. The Shift format enables a sample/shift register. The accumulator is loaded with an initial value from data on the LD port when the signal ACCUMSLOAD is toggled.



Estimated Resource Usage:
DSP_9x9: 4

Configuration

Size of the DSPMMAC block

	Width		Data Type	Source
Input A0/A1	18	(2-72)	Signed	Parallel
Input B0/B1	18	(2-72)	Signed	Parallel
Accumulator	52			

☐ Select Shift Out A ☐ Select Shift Out B

Operation: ACCUM+MULTA+MULTB

Register Options

Reset Mode: [✓ SYNC] ASYNC

	Clock	CE	RST
☑ Enable Input Register A	CLK0	CE0	RST0
☑ Enable Input Register B	CLK0	CE0	RST0
☑ Enable Pipeline Register	InputB	CE0	InputB
☑ Enable Output Register	Pipeline	CE0	Pipeline

Bus Ordering Style: Big Endian [MSB:LSB]

☐ Import LPC to ispLEVER project

Figure 3.3. MMAC Mode

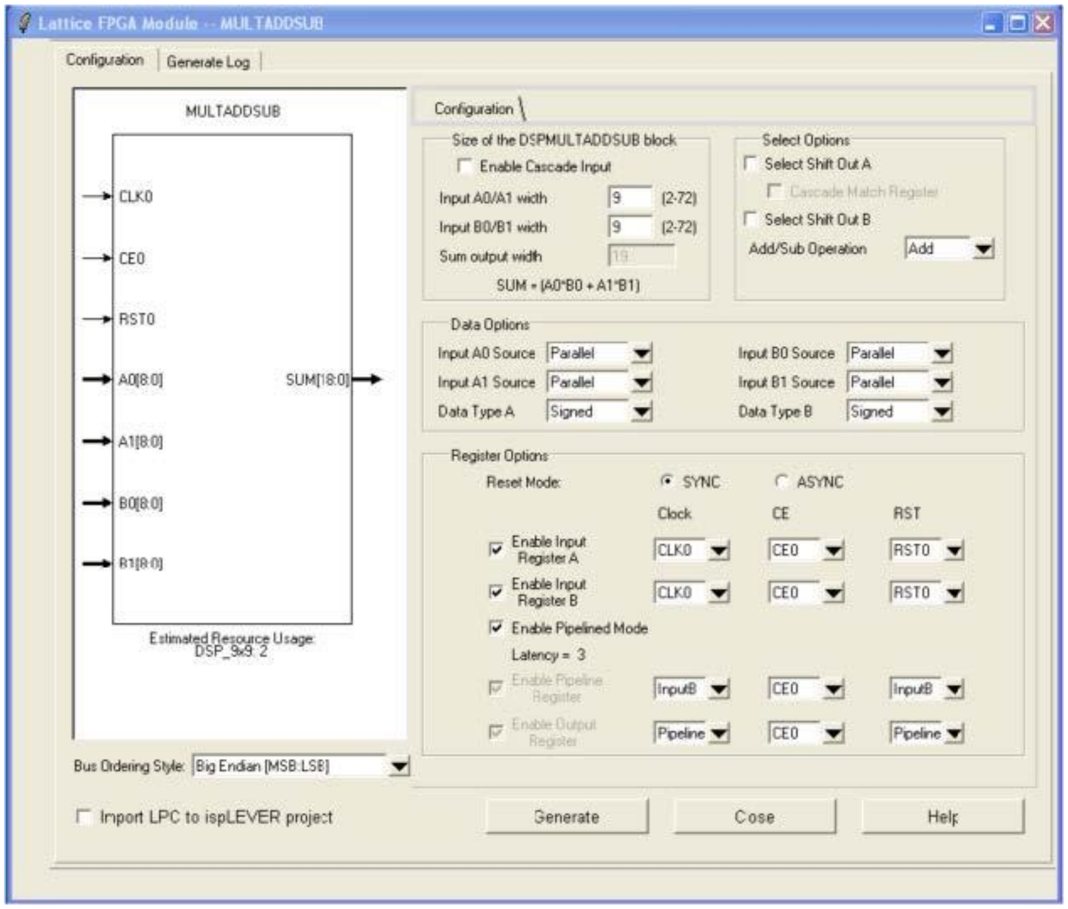
MULTADDSUB Module

The IPexpress MULTADDSUB Module configures elements to be packed into the sysDSP slice. The function A0 x B0 +/- A1 x B1 = P is implemented. The screen shot shown in Figure 3.4 shows the following:

- **Data Options**
  - Input bus widths from 2 to 72 bits
  - Cascade input can be enabled. This can be used to cascade the Multaddsub modules. If this is selected, the CIN and SignCIN will appear as additional input ports and SignSUM as an additional output port.
  - Data Type specifies whether the data is Signed, Unsigned or Dynamic
  - Source specifies whether data is from Parallel, Shift or Dynamic
  - Shift Out enables the shift out port on the sysDSP slice. The cascade match register is available if Shift Out A is selected. This is useful when a cascaded chain of Multaddsub modules are implemented.
  - Add/Sub Operation selects whether the arithmetic operation is Addition, Subtraction or Dynamic
- **Register Options**
  - Reset Mode selects if an Async or Sync Reset is used.

- Enable Input Register selects whether data input registers are used. If enabled, clock, reset and clock enable resources can be selected.
- Enable Pipeline Register selects whether data pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected. If the option InputA or InputB is selected, the pipeline register uses the same clock as the input register. Output registers will automatically be enabled and will use the same clock.
- Enable Output Register selects if the data output pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected.
- Enable Pipelined Mode turns on all the input, pipeline and output registers. If this mode is enabled, latency will be provided.

Multiple sysDSP slices can be spanned to accommodate large functions. Additional LUTs may be required if multiple sysDSP slices are needed. The input data format can be selected as Parallel, Shift or Dynamic. The Shift format can only be enabled if inputs are less than 18 bits. The Shift format enables a sample/shift register. Other than this shift register chain, the cascade chain can be enabled by selecting Enable cascade input, as described above. For appropriate operations, SUM and SignSUM (other than the last module in the cascade chain) need to be connected to CIN and SignCIN of an adjacent module. For the first module in the cascade chain, the CIN and SignCIN ports need to be assigned a value of 0.


Lattice FPGA Module -- MULTADDSUB

Configuration | Generate Log

MULTADDSUB: CLK0, CE0, RST0, A0[8:0], A1[8:0], B0[8:0], B1[8:0] → SUM[18:0]

Estimated Resource Usage:
DSP_9x9: 2

Configuration

Size of the DSPMULTADDSUB block
- Enable Cascade Input: unchecked
- Input A0/A1 width: 9 (2-72)
- Input B0/B1 width: 9 (2-72)
- Sum output width: 19
- SUM = (A0*B0 + A1*B1)

Select Options
- Select Shift Out A: unchecked
- Cascade Match Register: unchecked
- Select Shift Out B: unchecked
- Add/Sub Operation: Add

Data Options
- Input A0 Source: Parallel
- Input A1 Source: Parallel
- Data Type A: Signed
- Input B0 Source: Parallel
- Input B1 Source: Parallel
- Data Type B: Signed

Register Options
- Reset Mode: (●) SYNC ( ) ASYNC

	Clock	CE	RST
☑ Enable Input Register A	CLK0	CE0	RST0
☑ Enable Input Register B	CLK0	CE0	RST0
☑ Enable Pipeline Register	InputB	CE0	InputB
☑ Enable Output Register	Pipeline	CE0	Pipeline

- ☑ Enable Pipelined Mode
- Latency = 3

Bus Ordering Style: Big Endian [MSB:LSB]

☐ Import LPC to ispLEVER project

Figure 3.4. MULTADDSUB Mode

MULTADDSUBSUM Module

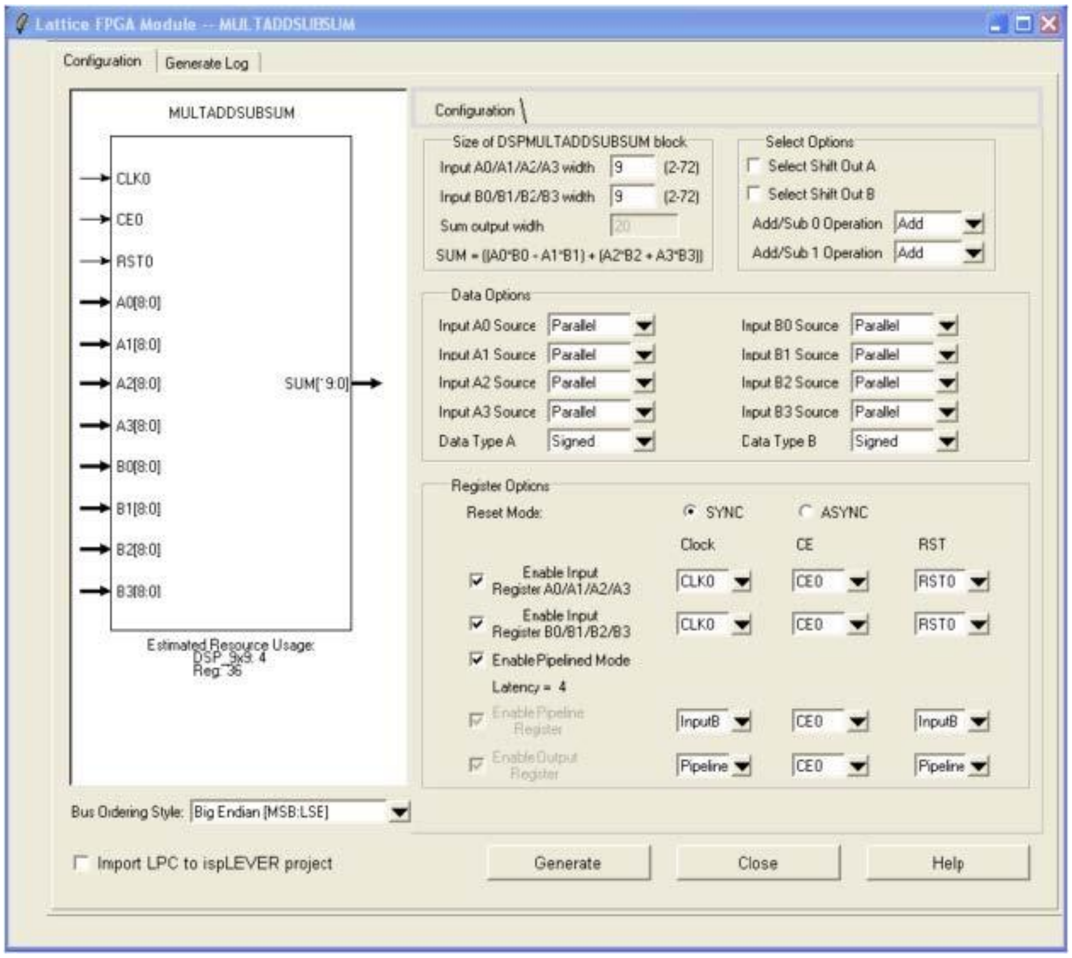
The IPexpress MULTADDSUBSUM Module configures elements to be packed into the sysDSP slice. The function A0 x B0 +/- A1 x B1 + A2 x B2 +/- A3 x B3 = P is implemented. The screen shot shown in Figure 3.5 shows the following:

- **Data Options**
  - Input bus widths from 2 to 72 bits
  - Data Type specifies whether the data is Signed, Unsigned or Dynamic

- Source specifies whether data is from Parallel, Shift or Dynamic
- Shift Out enables the shift out port on the sysDSP slice
- Add/Sub Operation selects whether the arithmetic operation is Addition, Subtraction or Dynamic

- **Register Options**
  - Reset Mode selects whether an Async or Sync Reset is used
  - Enable Input Register selects the data input registers to be used. If enabled, clock, reset and clock enable resources can be selected.
  - Enable Pipeline Register selects if data pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected. If the option Input A or Input B is selected, the pipeline register uses the same clock as the input register. Output registers will automatically be enabled and will use the same clock.
  - Enable Output Register selects whether the data output pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected.
  - Enable Pipelined Mode turns on all the input, pipeline and output registers. If this mode is enabled, latency will be provided.

Multiple sysDSP slices can be spanned to accommodate large functions. Additional LUTs may be required if multiple sysDSP slices are needed. The input data format can be selected as Parallel, Shift or Dynamic. The Shift format can only be enabled if inputs are less than 18 bits. The Shift format enables a sample/shift register.



Estimated Resource Usage:
DSP_9x9: 4
Reg: 36

Size of DSPMULTADDSUBSUM block
Input A0/A1/A2/A3 width: 9 (2-72)
Input B0/B1/B2/B3 width: 9 (2-72)
Sum output width: 20
SUM = ((A0*B0 - A1*B1) + (A2*B2 + A3*B3))

Select Options
☐ Select Shift Out A
☐ Select Shift Out B
Add/Sub 0 Operation: Add
Add/Sub 1 Operation: Add

Data Options

Input A0 Source	Parallel	Input B0 Source	Parallel
Input A1 Source	Parallel	Input B1 Source	Parallel
Input A2 Source	Parallel	Input B2 Source	Parallel
Input A3 Source	Parallel	Input B3 Source	Parallel
Data Type A	Signed	Data Type B	Signed

Register Options
Reset Mode: [✓ SYNC] ASYNC

	Clock	CE	RST
☑ Enable Input Register A0/A1/A2/A3	CLK0	CE0	RST0
☑ Enable Input Register B0/B1/B2/B3	CLK0	CE0	RST0
☑ Enable Pipelined Mode			
Latency = 4			
☑ Enable Pipeline Register	InputB	CE0	InputB
☑ Enable Output Register	Pipeline	CE0	Pipeline

Bus Ordering Style: Big Endian [MSB:LSB]
☐ Import LPC to ispLEVER project

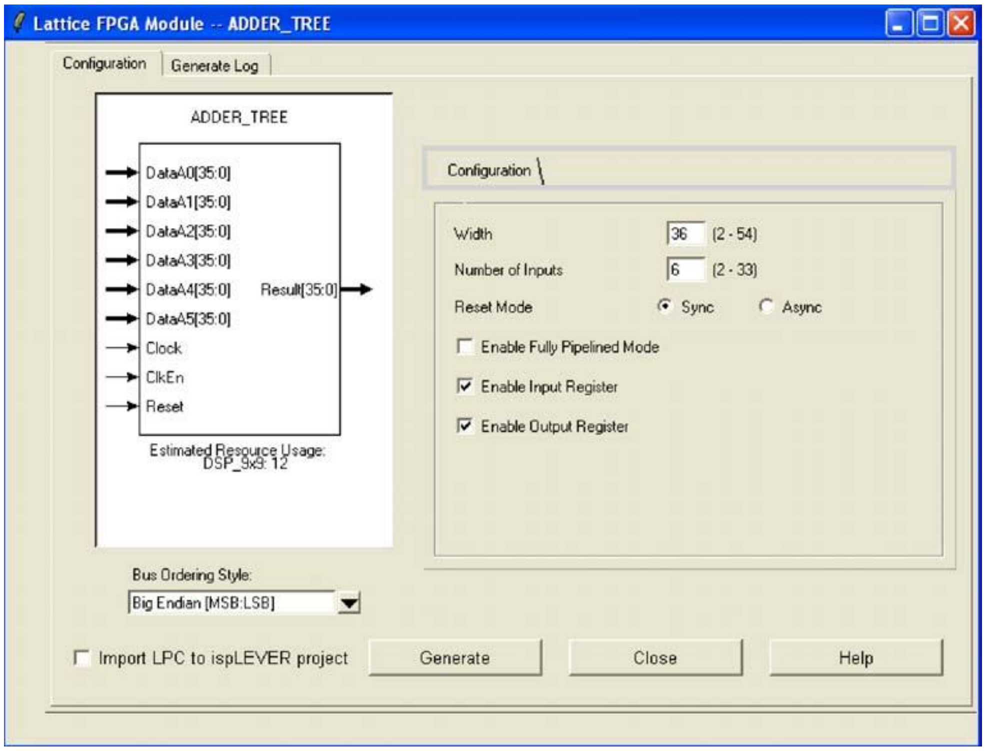
Figure 3.5. MULTADDSUBSUM Mode

Adder Tree Module

The IPexpress Adder Tree Module is used to generate an adder tree in the sysDSP slice. The function DataA0 + DataA1 + ... + DataAn = Resultn is implemented. The screen shot shown in Figure 3.6 shows the following:

- **Data Options**
  - Input bus widths from 2 to 54 bits
  - Number of inputs can be from 2 to 33
- **Register Options**
  - Reset Mode selects whether an Async or Sync Reset is used
  - Enable Fully Pipelined Mode selects if data pipeline and output registers are used
  - Enable Input Register selects if data input registers are used
  - Enable Output Register selects if data output registers are used

If the number of inputs is larger than 3, multiple DSP slices will be needed. Furthermore, if Fully Pipelined Mode is enabled, registers from the generic logic will be needed.



Estimated Resource Usage:
DSP_9x9: 12

Width 36 (2 - 54)
Number of Inputs 6 (2 - 33)
Reset Mode [✓ Sync] Async
☐ Enable Fully Pipelined Mode
☑ Enable Input Register
☑ Enable Output Register

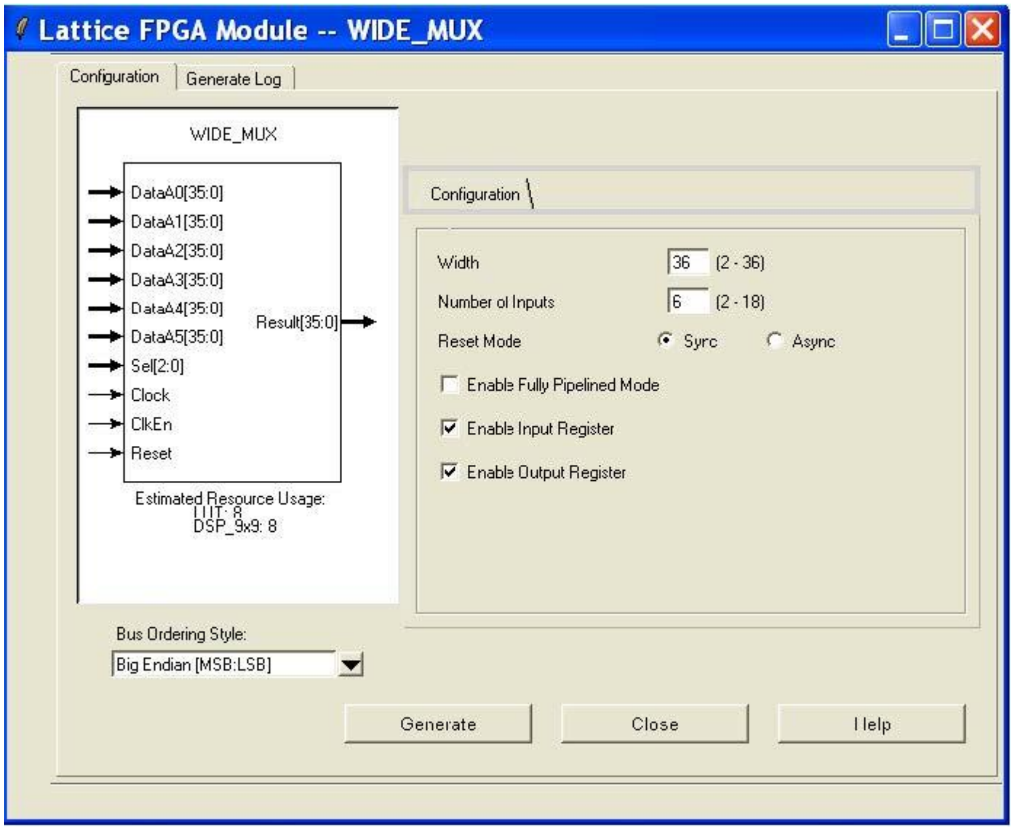
Bus Ordering Style: Big Endian [MSB:LSB]

Figure 3.6. Adder Tree Mode

Wide Mux Module

The IPexpress Wide Mux Module is used to generate a MUX in the sysDSP slice. The screen shot shown in Figure 3.7 shows the following:

- **Data Options**
  - Input bus widths from 2 to 36 bits
  - Number of inputs can be from 2 to 28
- **Register Options**
  - Reset Mode selects whether an Async or Sync Reset is used
  - Enable Fully Pipelined Mode selects if data input, pipeline and output registers are used
  - Enable Input Register selects if data input registers are used
  - Enable Output Register selects if data output registers are used.



Estimated Resource Usage:
LUT: 8
DSP_9x9: 8

Configuration

Width: 36 (2 - 36)
Number of Inputs: 6 (2 - 18)
Reset Mode: [✓ Sync] Async
☐ Enable Fully Pipelined Mode
☑ Enable Input Register
☑ Enable Output Register

Bus Ordering Style: Big Endian [MSB:LSB]

Figure 3.7. Wide Mux Mode

Barrel Shifter Module

The IPexpress Barrel Shifter Module is used to generate a Barrel Shifter in the sysDSP slice. Barrel Shifters are useful for applications such as floating point addition, compression/decompression algorithms, and pattern matching. The screen shot shown in Figure 3.8 shows the following:

- **Data Options**
    - Shift direction options of Left or Right
    - Type options of Zero Insert, Sign Extension (Shift Right only) or Rotate
    - Data Width can be from 1 to 40 bits or 2 to 32 for Type Rotate
- **Register Options**
    - Reset Mode selects whether an Async or Sync Reset is used
    - Enable Input Register selects if data input registers are used
    - Enable Pipeline Register selects if data pipeline registers are used
    - Enable Output Register selects if data output registers are used

Below are some examples of a Barrel Shifter:

Example 1:

Shift Direction = Left

Type = Zero Insert

A[7:0] = 0000 0100

Shift[1:0] = 10

P[7:0] = 0000 1000

Example 2:

Shift Direction = Right

Type = Sign Extension

A[7:0] = 1111 1100

Shift[1:0] = 10

P[7:0] = 1111 1110

Example 3:

Shift Direction = Left

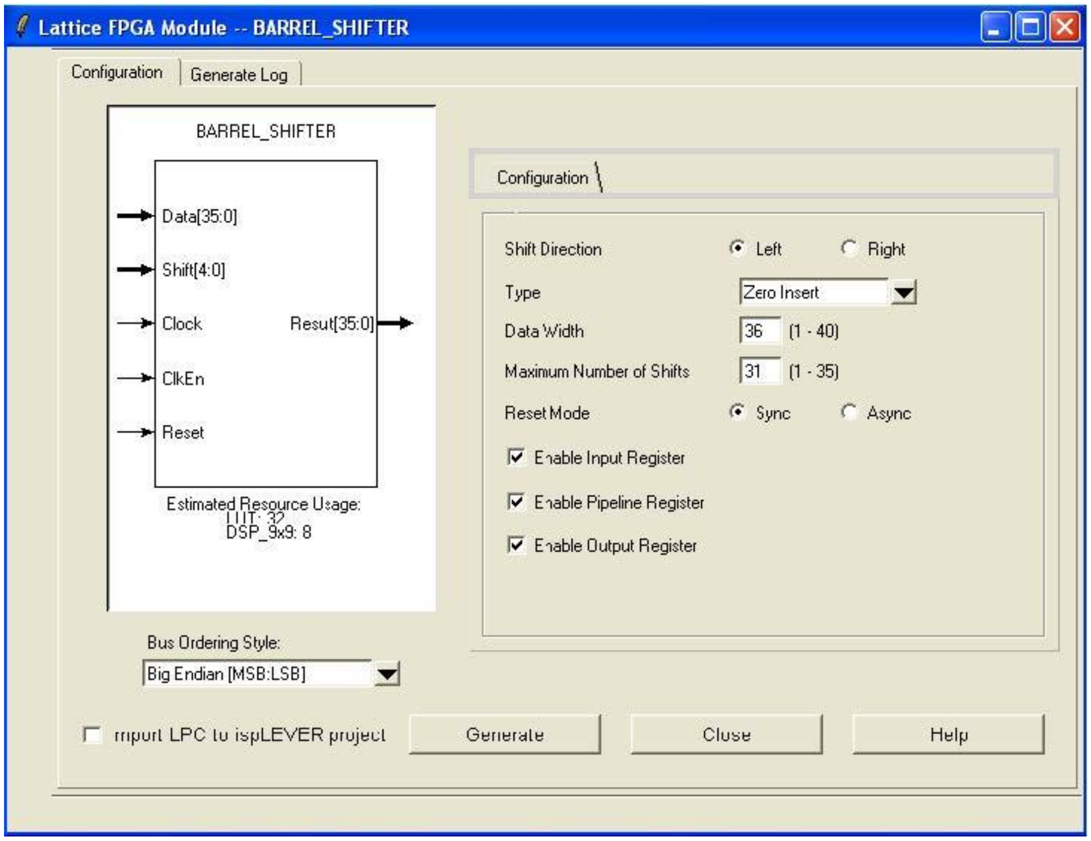
Type = Rotate

A[7:0] = 1111 1111 1100

Shift[1:0] = 10

P[7:0] = 1111 1111 1001

Multiple sysDSP slices can be spanned to accommodate large functions. Additional LUTs may be required if multiple sysDSP slices are needed.



Estimated Resource Usage:
LUT: 32
DSP_9x9: 8

Shift Direction: [✓ Left] Right
Type: Zero Insert
Data Width: 36 (1 - 40)
Maximum Number of Shifts: 31 (1 - 35)
Reset Mode: [✓ Sync] Async
[✓] Enable Input Register
[✓] Enable Pipeline Register
[✓] Enable Output Register

Bus Ordering Style: Big Endian [MSB:LSB]

[ ] Import LPC to ispLEVER project

Figure 3.8. Barrel Shifter Mode

3.3. Targeting the sysDSP Slice by Inference

The Inferencing flow enables the design tools to infer sysDSP slices from an HDL design. It is important to note that when using the Inferencing flow, unless the code style matches the sysDSP slice, results will not be optimal. The following are VHDL and Verilog examples.

VHDL Example

```
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

```
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
```

3.4. Targeting the sysDSP Slice Using ispLeverDSP with Simulink

Simulink is a graphical add-on (similar to schematic entry) for MATLAB, which is produced by The MathWorks. For more information, refer to the Simulink web page at www.mathworks.com/products/simulink/.

The ispLeverDSP is a block set in Simulink library provided by Lattice. It allows users to create algorithms and models in Simulink and converts the algorithms and models into RTL code.

After successful installation of ispLEVER or Diamond, set a path in MATLAB to make the Lattice block set available in the Simulink library. For more information, refer to the ispLEVER 8.1 Installation Notice or the Lattice Diamond Installation Notice.

The Lattice block set includes multipliers, adders, registers, and many other building blocks. Besides the basic building blocks there are a couple of unique Lattice blocks:

- **Gateways In and Out** – The design is made of the blocks between Gateway In and Gateway Out and these blocks can be converted into HDL code. When the design is converted into RTL code, the signal fed into the Gateway In is translated into the test vector for the testbench and the signal coming out of the Gateway Out is the reference copy for self-check in the testbench. In Figure 3.9, the input_x blocks on the left are Gateway In and the output block on the right is the Gateway Out.
- **Generator** – The Generator block is used to convert the design into HDL files that can be instantiated in a HDL design. The Generate Block is identified by the Lattice logo as shown in Figure 3.9.

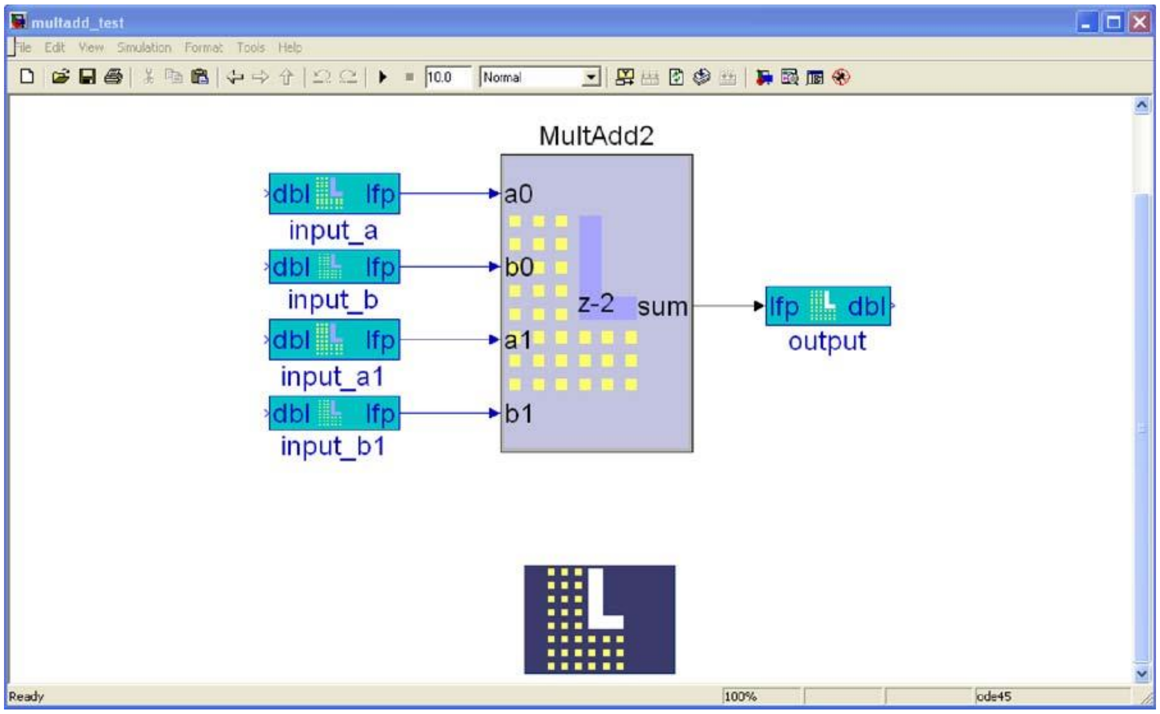


Figure 3.9. Simulink Design

3.5. Targeting the sysDSP Slice by Instantiating Primitives

The sysDSP slice can be targeted by instantiating the sysDSP slice primitives into a design. The advantage of instantiating primitives is that it provides access to all ports and sets all available parameters. The disadvantage of this flow is that all this customization requires extra coding by the user. Appendix A. DSP Primitives details the syntax for the sysDSP slice primitives.

4. sysDSP in the Report Files

4.1. Map Report

The Map Report includes information on how many sysDSP components are used and how many are available. A sysDSP slice is made of Multipliers and ALUs. The Map Report also shows how the sysDSP components are configured. Below is the DSP section from the Map Report Summary and the Component Details for the ALU.

```
Number Of Mapped DSP Components:
--------------------------------
MULT18X18C 2
MULT9X9C 0
ALU54A 1
ALU24A 0
--------------------------------
Number of Used DSP MULT Sites: 4 out of 256 (1 %)
Number of Used DSP ALU Sites: 2 out of 128 (1 %)
Number of clocks: 1
Net CLK0_c: 3 loads, 3 rising, 0 falling (Driver: PIO CLK0 )
DSP Component Details
---------------------
. ALU54A dsp_alu_0:
```

```
54-Bit ALU
Opcode 0 1
Opcode 1 0
Opcode 2 1
Opcode 3 0
Opcode 4 0
Opcode 5 0
Opcode 6 0
Opcode 7 0
Opcode 8 0
Opcode 9 1
Opcode 10 0
OpcodeOP0 Registers CLK CE RST
--------------------------------------------
Input CLK0 CE0 RST0
Pipeline CLK0 CE0 RST0
OpcodeOP1 Registers CLK CE RST
--------------------------------------------
Input -- --
Pipeline -- --
OpcodeIN Registers CLK CE RST
--------------------------------------------
Input
Pipeline
Data
Input Registers CLK CE RST
--------------------------------------------
C0
C1
Output Register CLK CE RST
--------------------------------------------
Output0 CLK0 CE0 RST0
Output1 CLK0 CE0 RST0
Flag Register CLK CE RST
Flag CLK0 CE0 RST0
Other
MCPAT_SOURCE STATIC
MASKPAT_SOURCE STATIC
MASK01 0x00000000000000
MCPAT 0x00000000000000
MASKPAT 0x00000000000000
RNDPAT 0x00000000000000
PSE17 0b11111111111111111
PSE44 0b11111111111111111111111111
PSE53 0b11111111
GSR DISABLED
RESETMODE SYNC
MULT9_MODE DISABLED
```

4.2. PAR Report

The PAR Report shows how the sysDSP components are packed into the sysDSP slices. The PAR Report shown below has two MULT18x18s and one ALU that are packed into one sysDSP slice:

```
---------------------------------- DSP Report ----------------------------------
DSP Slice #: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32
# of MULT9X9C
# of MULT18X18C
# of ALU24A
# of ALU54A
DSP Slice #: 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54
55 56 57 58 59 60 61 62 63 64
# of MULT9X9C
# of MULT18X18C 2
# of ALU24A
# of ALU54A 1
DSP Slice 33 Component_Type Physical_Type Instance_Name
MULT18_R52C2 MULT18X18C MULT18 dsp_mult_1
MULT18_R52C3 MULT18X18C MULT18 dsp_mult_0
ALU54_R52C5 ALU54A ALU54 dsp_alu_0
------------------------------ End of DSP Report ------------------------------
```

4.3. Trace Report

The Trace Report includes the timing of the design. The timing paths are analyzed to, from or through the sysDSP slice. Below is an example from the Post PAR Trace Report.

```
================================================================================
Preference: FREQUENCY NET "CLK0_c" 350.000000 MHz ;
72 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------
Passed: The following path meets requirements by 0.320ns
Logical Details: Cell type Pin type Cell/ASIC name (clock net +/-)
Source: MULT18X18C Port dsp_mult_0(ASIC) (from CLK0_c +)
Destination: ALU54A Port dsp_alu_0(ASIC) (to CLK0_c +)
Delay: 0.340ns (100.0% logic, 0.0% route), 1 logic levels.
Constraint Details:
0.340ns physical path delay dsp_mult_0 to dsp_alu_0 meets
2.857ns delay constraint less
0.000ns skew and
2.197ns MU_SET requirement (totaling 0.660ns) by 0.320ns
Physical Path Details:
Name Fanout Delay (ns) Site Resource
C2OUT_DEL --- 0.340 *18_R52C3.CLK0 to *T18_R52C3.P35 dsp_mult_0 (from CLK0_c)
ROUTE 1 0.000 *T18_R52C3.P35 to *54_R52C5.MB35 test_ecp3_mult_out_p_1_35 (to
CLK0_c)
--------
0.340 (100.0% logic, 0.0% route), 1 logic levels.
Clock Skew Details:
Source Clock:
Delay Connection
0.778ns N4.PADDI to MULT18_R52C3.CLK0
Destination Clock :
Delay Connection
```

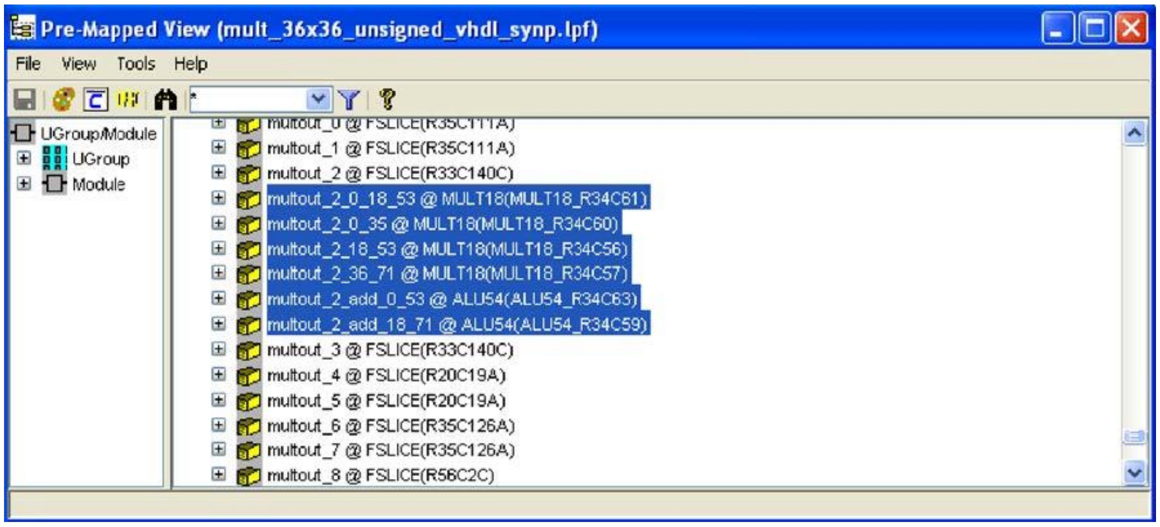
```
0.778ns N4.PADDI to ALU54_R52C5.CLK0
Report: 394.166MHz is the maximum frequency for this preference.
```

5. sysDSP in the ispLEVER Design Planner

Note: See Appendix B. Using IPexpress for Diamond for usage information for Lattice Diamond design software.

5.1. Pre-Mapped View

In the Design Planner Pre-Mapped View, sysDSP instances can be viewed or grouped together. Figure 5.1 is a screen shot showing MULT and ALU instances.



```
multout_0 @ FSLICE(R35C111A)
multout_1 @ FSLICE(R35C111A)
multout_2 @ FSLICE(R33C140C)
multout_2_0_18_53 @ MULT18(MULT18_R34C61)
multout_2_0_35 @ MULT18(MULT18_R34C60)
multout_2_18_53 @ MULT18(MULT18_R34C56)
multout_2_36_71 @ MULT18(MULT18_R34C57)
multout_2_add_0_53 @ ALU54(ALU54_R34C63)
multout_2_add_18_71 @ ALU54(ALU54_R34C59)
multout_3 @ FSLICE(R33C140C)
multout_4 @ FSLICE(R20C19A)
multout_5 @ FSLICE(R20C19A)
multout_6 @ FSLICE(R35C126A)
multout_7 @ FSLICE(R35C126A)
multout_8 @ FSLICE(R56C2C)
```

Figure 5.1. Design Planner Pre-Mapped View

5.2. Floorplan View

The DSP slices are organized in rows, as shown in Figure 5.2. It can be seen that each slice extends to four columns with the multipliers on the left and ALUs on the right.

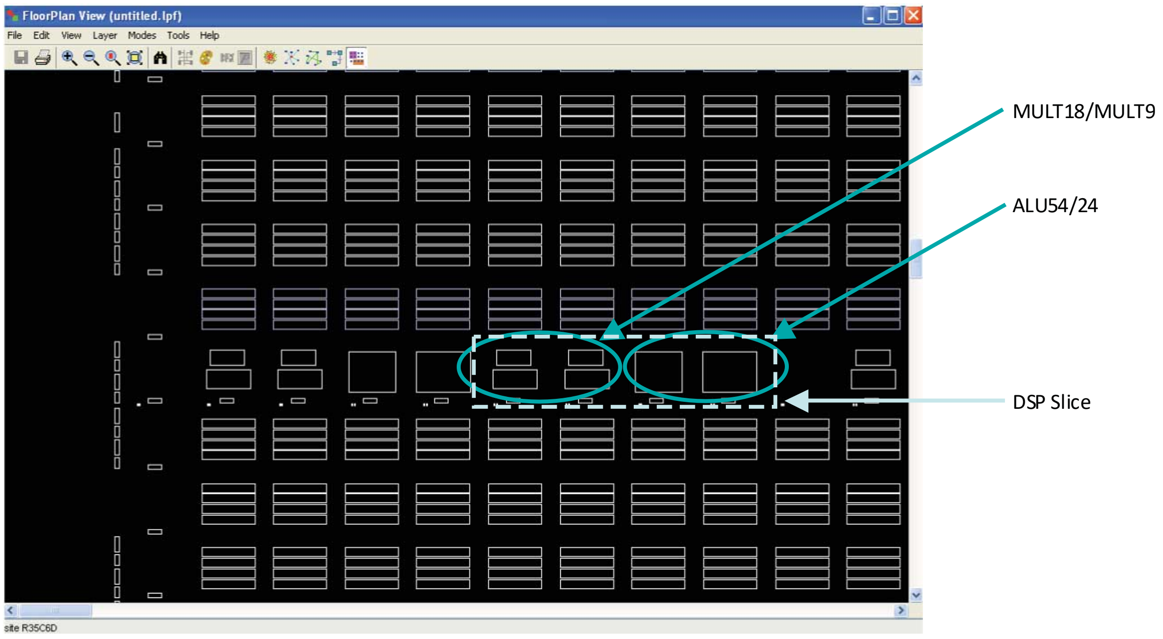
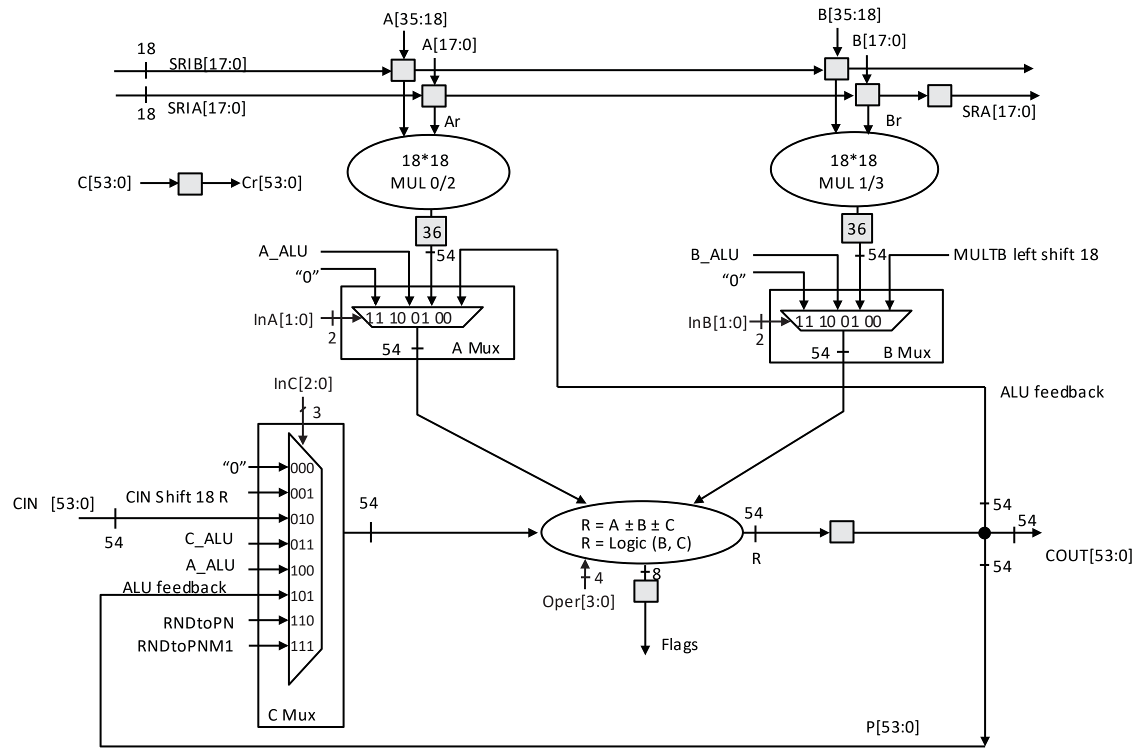

FloorPlan View (untitled.lpf)
File Edit View Layer Modes Tools Help
MULT18/MULT9
ALU54/24
DSP Slice
site R35C6D

Figure 5.2. Floorplan View

6. Advanced Features of the sysDSP Slice

Examining the LatticeECP3 sysDSP slice in more detail (Figure 6.1) reveals the Arithmetic Logic Unit (ALU) and three programmable muxes, AMUX, BMUX and CMUX. These components are used in combination to enable the advanced functions of the sysDSP slice, such as:

- Cascading of slices for implementing Adder Trees fully in sysDSP slices.
- Ternary addition functions are implemented through bypassing of multipliers.
- Various rounding techniques modify the data using the ALU.
- ALU flags.
- Dynamic muxes input selection allows for Time Division Multiplexing (TDM) of the sysDSP slice resources.



Notes:
1. Two slices are shown: Slice0 (Mult0 and Mult1) and Slice1 (Mult2 and Mult3)
2. Each 18x18 can also implement two 9x9s.
3. A Mux[53:36] and B Mux[53:36] are sign extended if SIGNEDA or SIGNEDB = 1; otherwise, they are zero extended.
4. P[17:0] has an independent multiplexer to bypass or not bypass the COUT[17:0] register. This is required to support 36*36 multiplication mode.
5. A 54-bit ALU does not always feed 54 bits from A, B and C. Sometimes it will select only 36b, as seen in "Arithmetic Modes".

▢ = Bypassable registers

Figure 6.1. Detailed sysDSP Slice Diagram

AMUX

AMUX selects between multiple 54-bit inputs to the ALU statically or dynamically. The inputs to AMUX are listed in Table 6.1.

Table 6.1. AMUX Inputs

InA[1:0]	AMUX Signal Selection
00	ALU feedback
01	Output of MULTA, sign-extended to 54 bits
10	A_ALU={Cra[17:0], Ara}, concatenated result of Cra and Ara
11	GND or 54 bits of 0

Signal Descriptions:

- Cra is generated from Cr by sign extending Ar35 (on Cr(17:0)), and Br35 (on Cr(44:27))
- Ara is generated from the registered input of A (AR) sign extended to 36 bits

BMUX

BMUX selects between multiple 54-bit inputs to the ALU statically or dynamically. The inputs to BMUX are listed in Table 6.2.

Table 6.2. BMUX Inputs

InB[1:0]	BMUX Signal Selection
00	Output of MULTB, left shift 18 bits
01	Output of MULTB, sign-extended to 54 bits
10	B_ALU={Cra[44:27], Bra}, concatenated result of Cra and Bra
11	GND or 54 bits of 0

Signal Descriptions:

- Bra is generated from the registered input of B (BR) sign extended to 36 bits

CMUX

CMUX selects between multiple 54-bit inputs to the ALU statically or dynamically. The inputs to CMUX are listed in Table 6.3.

Table 6.3. CMUX Inputs

InC[2:0]	CMUX Signal Selection
000	GND or 54 bits of 0
001	CIN right shift 18 bits
010	CIN
011	C_ALU: sign extended Cr
100	A_ALU
101	ALU feedback
110	Constant for rounding (RNDtoPN)
111	Constant for rounding (RNDtoPNM1)

Signal Descriptions:

- CIN (Cascade Input) is connected to the COUT (Cascade Output) of an adjacent sysDSP slice

ALU

Operation Codes: The ALU does the processing of the sysDSP slice. Its operation can be static or dynamic and is set using a 4-bit operation code. The ALU can operate with either two inputs (BMUX and CMUX) or with three inputs mode (AMUX, BMUX and CMUX). Table 6.4 lists the operation codes and the ALU functions that are performed.

Table 6.4. Operation Codes and ALU Functions

#	OPR(3)	OPR(2)	OPR(1)	OPR(0)	Function	
1	0	1	0	0	R = A + B + C	Arithmetic
2	0	1	0	1	R = A - B + C	
3	0	1	1	0	R = A + B - C	
4	0	1	1	1	R = A - B - C	
5	1	1	0	0	R = B XNOR C	Logical
6	1	1	1	0	R = B XOR C	
7	**0**	**0**	**0**	**0**	**R = B NAND C**	
8	1	0	0	0	R = B AND C	
9	0	0	1	1	R = B OR C	
10	1	0	1	1	R = B NOR C	

Signal Descriptions:

- OPR(x) is the opcode bit number
- A is the output from the AMUX
- B is the output from the BMUX
- C is the output from the CMUX

For arithmetic operation, the ALU is a ternary adder with selectable addition or subtraction for each input. The data width for the ALU operation is 54 bits with the following exception: MUX A and B opcodes are 10 and MUX C opcode is 011. For this operation mode, the data width is limited to 36 bits. If the 0 input port for one of the MUXes is selected by the port selection port, the ALU function effectively becomes a two-input operation.

Flags: ALU flags are indicators generated by comparing the ALU result and constants or fixed patterns Figure 6.2 and Figure 6.3 show the circuitry for the flags. Table 6.5 and Table 6.6 provide definitions of the inputs and outputs.

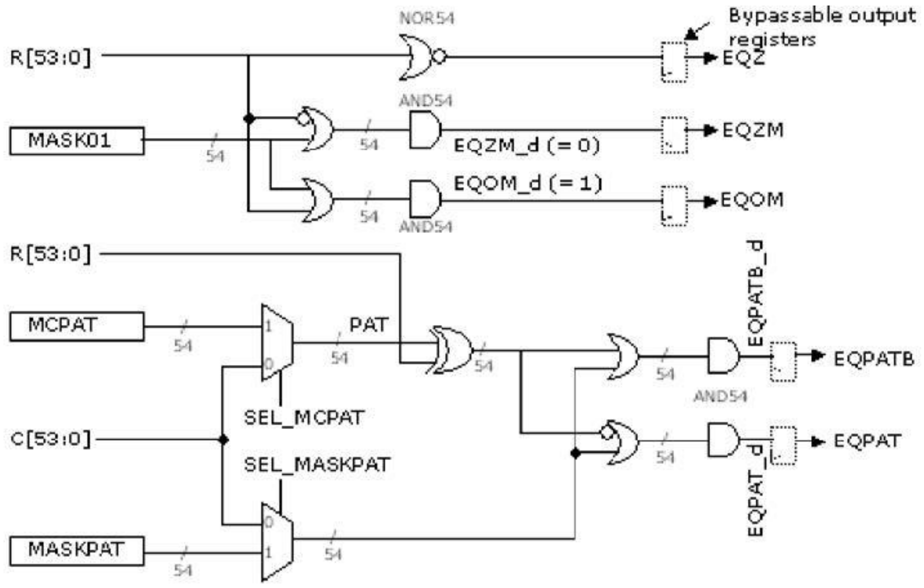


Figure 6.2. Flag Circuitry

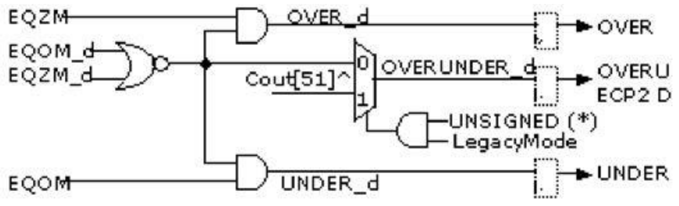


Figure 6.3. Flag Pattern Circuitry

Table 6.5. Flag Input Definitions

Flag Input	Definition
R	54-bit result from the ALU
MASK01	Memory cell constant which is a mask for EQZM/EQOM
MASKPAT	Memory cell constant which is a mask constant for EQPAT/EQPATB
MCPAT	Memory cell constant which is a MEM Cell Pattern constant
SELPAT	Mux select chooses the PAT source
SELMASK	Mux select chooses the mask source for EQPAT/EQPATB
LegacyMode	Sets OVERUNDER which is used to support LatticeECP2 legacy overflow

Table 6.6. Flag Output Definitions

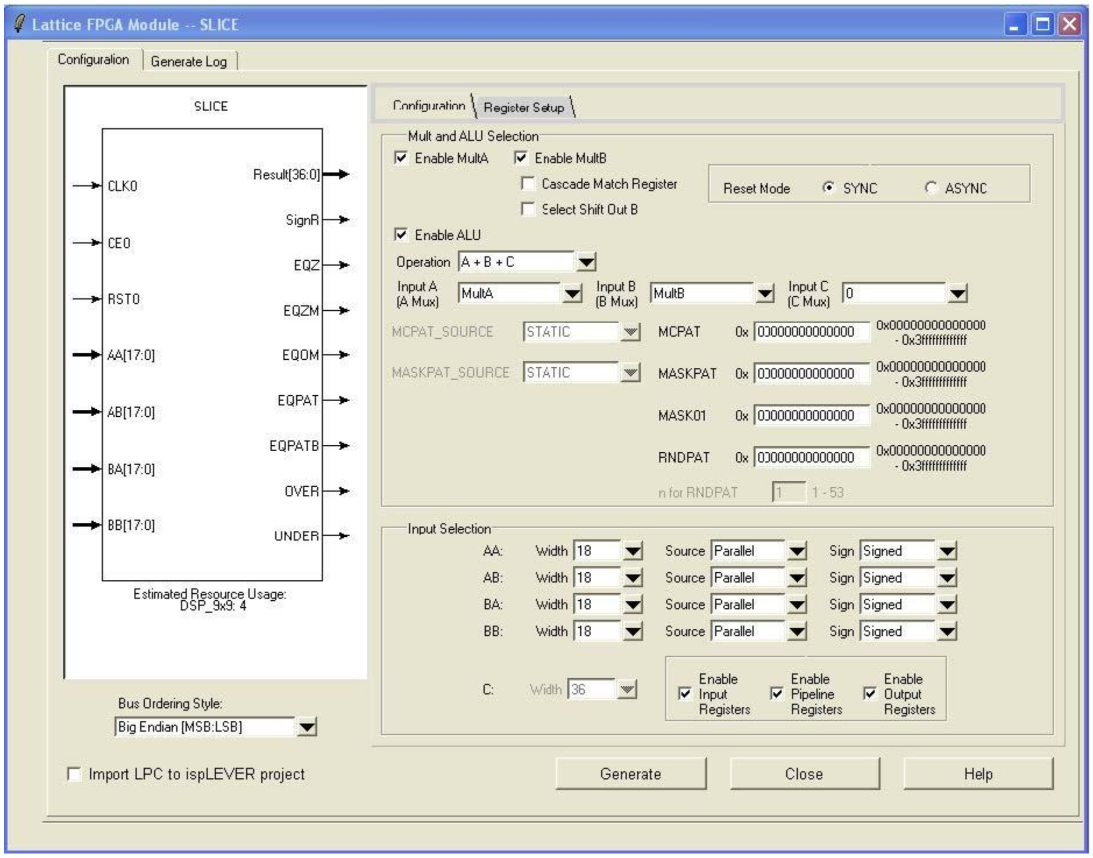
Flag Output	Definition	Equation
EQZ	Equal to zero	EQZ = not(bitwise_or(R))
EQZM	Equal to zero with mask	EQZM = bitwise_and(not(R) or MASK01)
EQOM	Equal to one with mask	EQOM = bitwise_and(R or MASK01)
EQPAT[1]	Equal to PAT with mask	EQPAT = bitwise_and(not(R xor MCPAT) or MASKPAT)
EQPATB[1]	Equal to bit inverted PAT with mask	EQPATB = bitwise_and((R xor MCPAT) or MASKPAT)
OVER	Accumulator overflow	OVER = EQZM and (not(EQOM_d or EQZM_d))
UNDER	Accumulator underflow	UNDER = EQOM and (not(EQOM_d or EQZM_d))
OVERUNDER	Either over or underflow	
_d	The suffix "_d" denotes non-registered signals	

Note:

1. For EQPAT and EQPATB, the pattern to be compared with can be specified through MCPAT and MASKPAT or through C input of sysDSP slice dynamically.

7. IPexpress Slice Module

See Appendix B. Using IPexpress for Diamond for information on using the IPexpress Slice Module in Diamond. The IPexpress Slice Module allows users to configure a sysDSP slice and enable all of its advanced features. Figure 7.1 shows the GUI for the Slice Module.



Estimated Resource Usage:
DSP_9x9: 4

Bus Ordering Style: Big Endian [MSB:LSB]

☐ Import LPC to ispLEVER project

Mult and ALU Selection

☑ Enable MultA ☑ Enable MultB

☐ Cascade Match Register

☐ Select Shift Out B

Reset Mode (● SYNC ○ ASYNC)

☑ Enable ALU

Operation: A + B + C

Input A (A Mux): MultA   Input B (B Mux): MultB   Input C (C Mux): 0

MCPAT_SOURCE: STATIC   MCPAT 0x 00000000000000   0x00000000000000 - 0x3fffffffffffff

MASKPAT_SOURCE: STATIC   MASKPAT 0x 00000000000000   0x00000000000000 - 0x3fffffffffffff

MASK01 0x 00000000000000   0x00000000000000 - 0x3fffffffffffff

RNDPAT 0x 00000000000000   0x00000000000000 - 0x3fffffffffffff

n for RNDPAT 1   1 - 53

Input Selection

	Width	Source	Sign
AA:	18	Parallel	Signed
AB:	18	Parallel	Signed
BA:	18	Parallel	Signed
BB:	18	Parallel	Signed
C:	36		

☑ Enable Input Registers   ☑ Enable Pipeline Registers   ☑ Enable Output Registers

Figure 7.1. Slice Module GUI

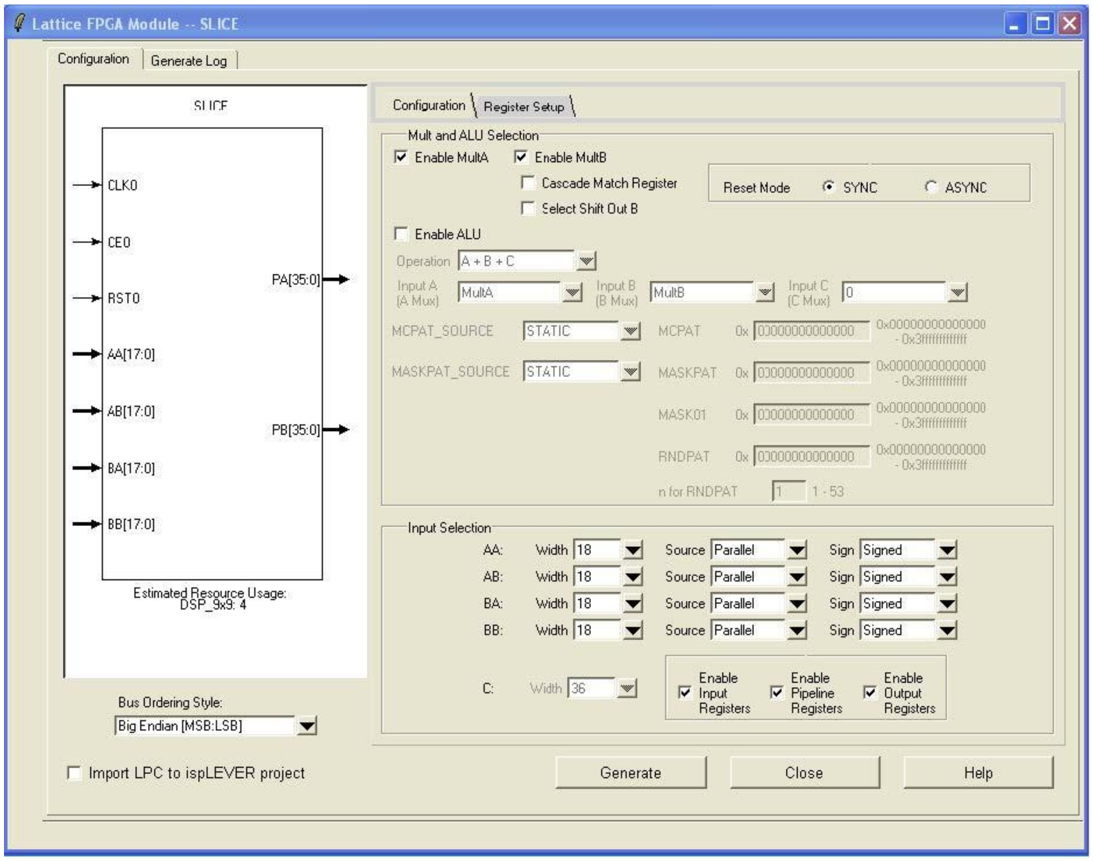
There are two sections for configuration, Mult/ALU and Input Selections. The output ports, such as data and flags, are enabled automatically depending on sysDSP operation and user options. The two sections are described below.

Input Selection

- Input bus widths which can be from 1-18 bit for AA, AB, BA, BB. AA and AB are the inputs for MultA, BA and BB are the inputs for MultB.
- Input bus widths which can be from 1-54 bit for C (only when "Enable B" is unchecked). This C input port is different from the CMUX output (input of ALU).
- Data Type specifies if the data is Signed, Unsigned or Dynamic
- Source specifies if data is from Parallel, Shift or Dynamic
- Shift Out enables the shift out port on the sysDSP slice
- Input, Pipeline and Output Registers are user-selectable

Mult and ALU Selection

To configure a sysDSP slice, the user must first consider whether the ALU is needed. The result of the ALU is a function of A, B and C, where A, B and C are outputs of AMUX, BMUX and CMUX, respectively. If the check box “Enable ALU” is unchecked, all the ALU related options will be grayed out and thus not available, as shown in Figure 7.2. These include Operation, Input A (A Mux), Input B (B Mux), Input C (C Mux), MCPAT_SOURCE, MASKPAT_SOURCE, MCPAT, MASKPAT, MASK01, and RNDPAT. This will set up the slice as two independent multipliers.



Estimated Resource Usage:
DSP_9x9: 4

Bus Ordering Style: Big Endian [MSB:LSB]

☐ Import LPC to ispLEVER project

Mult and ALU Selection

☑ Enable MultA ☑ Enable MultB
☐ Cascade Match Register
☐ Select Shift Out B
Reset Mode ● SYNC ○ ASYNC
☐ Enable ALU
Operation: A + B + C
Input A (A Mux): MultA Input B (B Mux): MultB Input C (C Mux): 0
MCPAT_SOURCE: STATIC MCPAT: 0x 00000000000000 (0x00000000000000 - 0x3fffffffffffff)
MASKPAT_SOURCE: STATIC MASKPAT: 0x 00000000000000 (0x00000000000000 - 0x3fffffffffffff)
MASK01: 0x 00000000000000 (0x00000000000000 - 0x3fffffffffffff)
RNDPAT: 0x 00000000000000 (0x00000000000000 - 0x3fffffffffffff)
n for RNDPAT: 1 (1 - 53)

Input Selection

	Width	Source	Sign
AA:	18	Parallel	Signed
AB:	18	Parallel	Signed
BA:	18	Parallel	Signed
BB:	18	Parallel	Signed
C:	36		

☑ Enable Input Registers ☑ Enable Pipeline Registers ☑ Enable Output Registers

Figure 7.2. Slice Module GUI with "Enable ALU" Unchecked

The check box “Enable ALU” should be checked if the ALU is needed. The ALU can operate in either three-input or two-input mode or it can be configured dynamically.

- **Three-input operation options**: A+B+C, A+B-C, A-B+C, A-B-C.
- **Two-input operation options**: B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C.
- **Dynamic option**: Any of the possible Opcodes listed in Table 6.4.

Next, determine whether MultA, MultB, or both, will be used.

For CMUX, the output can be one of the eight inputs or it can be configured dynamically. The available configurations for the outputs of AMUX and BMUX are dependent on whether MultA and MultB are enabled.

- If “Enable MultA” is checked, the output of AMUX can be either MultA output or configured dynamically.
- If “Enable MultA” is unchecked, the options are ALU feedback, A_ALU, 0 and DYNAMIC.
- If “Enable MultB” is checked, the output of BMUX can be MultB output, MultB shift 18L or configured dynamically.
- If “Enable MultB” is unchecked, the options are B_ALU, 0 and DYNAMIC.

Configuration of “Enable MultA” will affect the availability of the operation options:

- If “Enable MultA” is checked, the available options are A + B + C, A - B + C, A + B - C, A - B - C and DYNAMIC;
- If “Enable MultA” is unchecked, the available options are A + B + C, A - B + C, A + B - C, A - B - C, B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C and DYNAMIC.

Configuration of “Enable MultB” will affect the availability of the cascade match register and select shift out B.

- If “Enable MultB” is checked, “cascade match register” and “select shift out B” will be available. CascA0 and CascA1 in Figure 2.1 are controlled by the check box of cascade match register.
- If “cascade match register” is checked, registers CascA0 and CascA1 will be added to the line of shift register A.
- If “select shift out B” is checked, both SROA and SROB will be added to output ports of the Slice module. Figure 3.7 lists the possible configuration scenarios.

Table 7.1. Possible Configuration Scenarios

Scenario	MULTA	MULTB	ALU	Operation Mode
1	Yes	No	No	a. Single multiplier instance: MULTA only. b. Static mode only.[1]
2	No	Yes	No	a. Single multiplier instance: MULTB only. b. Static mode only.
3	Yes	Yes	No	a. Two multiplier instances: MULTA and MULTB b. Static mode only.
4	Yes	Yes	Yes	a. Two multiplier instances and ALU instance: MULTA, MULTB and ALU. b. Static mode: A + B + C or A - B + C or A + B - C or A - B - C. c. Dynamic mode[2]: A + B + C, A - B + C, A + B - C, A - B - C, B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C
5	No	Yes	Yes	a. One multiplier instance and ALU instance: MULTB and ALU. b. Static mode: A + B + C, A - B + C, A + B - C, A - B - C, B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C c. Dynamic mode: A + B + C, A - B + C, A + B - C, A - B - C, B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C.
6	Yes	No	Yes	a. One multiplier instance and ALU instance: MULTA and ALU. b. Static mode: A + B + C, A - B + C, A + B - C, A - B - C, B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C c. Dynamic mode: A + B + C, A - B + C, A + B - C, A - B - C, B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C
7	No	No	Yes	a. ALU instance: ALU only. b. Static mode: A + B + C, A - B + C, A + B - C, A - B - C, B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C c. Dynamic mode: A + B + C, A - B + C, A + B - C, A - B - C, B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C

Notes:

1. Static mode means available operation options other than DYNAMIC.
2. Dynamic mode means the operation can be configured dynamically.

If the ALU is used, some flag signals are generated by comparing the ALU results with pre-specified constants or C input. In the GUI, these constants are MASK01, MCPAT, and MASKPAT.

- If "Enable MultB" is unchecked, options for MCPAT_SOURCE and MASKPAT_SOURCE will be available. These options are STATIC and DYNAMIC.
- If "STATIC" is checked, the values for MCPAT and MASKPAT need to be specified by the user.
- If "DYNAMIC" is checked, the values for MCPAT and MASKPAT are specified though the input port C dynamically.
- If "Enable ALU" is checked, there will be an output port SignR available, which indicates the sign of the result. This port should be left unconnected if not being used.

8. Rounding

LatticeECP3 sysDSP slices provide the capability of rounding through the two rounding patterns. Rounding on a number is achieved by adding a constant (e.g. 0.5 for decimal) followed by a truncation, or in the following equation:

Output = Floor(Input + 0.5), with input and output in decimal

In sysDSP slices, there are two rounding constant inputs for CMUX. One is 0...0001000...0, (called RNDtoPN) with the binary or rounding point between the "1" and the "0" in front of the "1", and the value of RNDtoPN is 0.5 in decimal. The other one is 0...00001111...1, (called RNDtoPNM1) with the binary or rounding point between the last "0" and the first "1". Where N is the position of binary point or the number of 1s. By choosing one of these constants, the output of the sysDSP slice is (result of ALU)+ RNDtoPN or (result of ALU)+ RNDtoPNM1. The ALU result at the previous clock cycle is the input for the rounding operation and the rounding result is:

Output_truncated = Floor(Input + RNDtoPN)

or

Output_truncated = Floor(Input + RNDtoPNM1)

Other manipulations are also possible, depending on the rounding scheme to be implemented. Thus, these two constants are chosen by the user and it is up to the user to determine which rounding scheme is implemented. The information of the ALU output value (positive, negative etc.) can be extracted and used for port selection of CMUX, thus the selection of rounding constants.

The following are examples of rounding. Example 1 is described in detail and the others are similar.

8.1. Example 1: Rounding Toward Zero

To implement rounding to zero, for a positive input, RNDtoPNM1 is used; for a negative input, RNDtoPN is used. In the GUI, the value of RNDPAT needs to be specified by the user, which is used to specify the rounding point position (e.g. 000000 00000000 00000000 00000000 00000000 00000000 000 10000). The user is responsible for the switching of the RND_CONSTANTs depending on the sign of the ALU result and rounding scheme.

Steps for rounding toward zero:

1. Extract the sign of the ALU result.
2. Select the input port of the CMUX based on the result of step 1. If the sign is positive, RNDtoPNM1 will be selected to pass through CMUX. If the sign is negative, RNDtoPN will be selected to pass through CMUX.
3. Add the RND_CONSTANT and the ALU result.
4. Truncate the result of step 3. This is the rounding result.

Rounding to zero of a 54-bit value to 49-bit

Rounding point

Sign	Data Type	Data Value
+	Input	000000 00000000 00000000 00000000 00000000 01010101 010 10000
	RNDtoPNM1	000000 00000000 00000000 00000000 00000000 00000000 000 01111
	Output Data	000000 00000000 00000000 00000000 00000000 01010101 010 1 1111
	Output truncated	000000 00000000 00000000 00000000 00000000 01010101 010
-	Input	111111 11111111 11111111 11111111 11111111 01010101 010 10000
	RNDtoPN	000000 00000000 00000000 00000000 00000000 00000000 000 10000
	Output Data	111111 11111111 11111111 11111111 11111111 01010101 01 1 00000
	Output truncated	111111 11111111 11111111 11111111 11111111 01010101 011

8.2. Example 2: Rounding to Positive Infinity

Rounding to positive infinity of a 54-bit value to 49-bit

Rounding point

Sign	Data Type	Data Value
+	Input	000000 00000000 00000000 00000000 00000000 01010101 010 10000
	RNDtoPN	000000 00000000 00000000 00000000 00000000 00000000 000 10000
	Output Data	000000 00000000 00000000 00000000 00000000 01010101 01 1 00000
	Output truncated	000000 00000000 00000000 00000000 00000000 01010101 011
-	Input	111111 11111111 11111111 11111111 11111111 01010101 010 10000
	RNDtoPN	000000 00000000 00000000 00000000 00000000 00000000 000 10000
	Output Data	111111 11111111 11111111 11111111 11111111 01010101 01 1 00000
	Output truncated	111111 11111111 11111111 11111111 11111111 01010101 011

8.3. Example 3: Rounding to Negative Infinity

Rounding to negative infinity of a 54-bit value to 49-bit

Rounding point

Sign	Data Type	Data Value
+	Input	000000 00000000 00000000 00000000 00000000 01010101 010 10000
	RNDtoPNM1	000000 00000000 00000000 00000000 00000000 00000000 000 01111
	Output Data	000000 00000000 00000000 00000000 00000000 01010101 010 11111
	Output truncated	000000 00000000 00000000 00000000 00000000 01010101 010
-	Input	111111 11111111 11111111 11111111 11111111 01010101 010 10000
	RNDtoPNM1	000000 00000000 00000000 00000000 00000000 00000000 000 01111
	Output Data	111111 11111111 11111111 11111111 11111111 01010101 010 11111
	Output truncated	111111 11111111 11111111 11111111 11111111 01010101 010

8.4. Example 4: Rounding Away from Zero

Rounding away from zero of a 54-bit value to 49-bit

Rounding point

Sign	Data Type	Data Value
+	Input	000000 00000000 00000000 00000000 00000000 01010101 010 10000
	RNDtoPN	000000 00000000 00000000 00000000 00000000 00000000 000 10000
	Output Data	000000 00000000 00000000 00000000 00000000 01010101 01 1 00000
	Output truncated	000000 00000000 00000000 00000000 00000000 01010101 011
-	Input	111111 11111111 11111111 11111111 11111111 01010101 010 10000
	RNDtoPNM1	000000 00000000 00000000 00000000 00000000 00000000 000 01111
	Output Data	111111 11111111 11111111 11111111 11111111 01010101 010 11111
	Output truncated	111111 11111111 11111111 11111111 11111111 01010101 010

For convergent rounding, there are two options: one is rounding toward even and the other is rounding toward odd. For rounding toward even, the number of exact x.5 is rounding toward the closest even and the others are rounding toward the nearest.

In addition to the steps listed above for rounding toward zero or infinity, some extra steps are needed. If the value of (result of ALU) + RND_CONSTANT matches a certain pattern, or if flag EQPAT = 1 is generated, the least significant bit after truncation will be assigned 0 (rounding toward even) or 1 (rounding toward odd). For rounding toward even, MAPAT = 11...1100...00, MASKPAT = 11...1100...00, where the rounding point is between the last "1" and the first "0". The flag EQPAT generated with this pattern indicates whether the (result of ALU) + RND_CONSTANT matches a certain pattern of XX...X100...00.

8.5. Example 5: Rounding Toward Even

Rounding toward even steps (54-bit value to 49-bit MSBs, rounding point = 5)

1. Add RNDtoPN to the result of ALU
2. If EQPAT=1(matches pattern XX...X100...00), then integer LSB is replaced with 0; otherwise, keep the LSB.

Rounding toward even of a 54-bit value to 49-bit

Rounding toward even of a 54-bit value to 49-bit

Rounding point

Sign	Data Type	Data Value
+	Input	000000 00000000 00000000 00000000 00000000 00000000 010 10000
	RNDtoPN	000000 00000000 00000000 00000000 00000000 00000000 000 10000
	Output Data	000000 00000000 00000000 00000000 00000000 00000000 01 1 00000
	Match 100000	Yes
	Output truncated	000000 00000000 00000000 00000000 00000000 00000000 010
-	Input	111111 11111111 11111111 11111111 11111111 11111111 101 10000
	RNDtoPN	000000 00000000 00000000 00000000 00000000 00000000 000 10000
	Output Data	111111 11111111 11111111 11111111 11111111 11111111 1 10 00000
	Match 100000	No
	Output truncated	111111 11111111 11111111 11111111 11111111 11111111 110

8.6. Example 6: Rounding Toward Odd

For rounding toward odd, MCPAT = 11...11011...11, MASKPAT = 11...1100...00, where the rounding point is between the last "1" and the first "0". The flag EQPAT generated with this pattern indicates whether the (result of ALU) + RND_CONSTANT matches a certain pattern of XX...X011...11.

Rounding toward odd steps (rounding point = 5):

1. Add RNDtoPNM1 to the result of ALU
2. If EQPAT=1(matches pattern XX...X01...11), then integer LSB is replaced with 1; otherwise, keep the LSB.

Rounding toward odd of a 54-bit value to 49-bit

Rounding toward odd of a 54-bit value to 49-bit

Rounding point

Sign	Data Type	Data Value
+	Input	000000 00000000 00000000 00000000 00000000 00000000 010 10000
	RNDtoPNM1	000000 00000000 00000000 00000000 00000000 00000000 000 01111
	Output Data	000000 00000000 00000000 00000000 00000000 00000000 01 0 11111
	Match 011111	Yes
	Output truncated	000000 00000000 00000000 00000000 00000000 00000000 011
-	Input	111111 11111111 11111111 11111111 11111111 11111111 101 10000
	RNDtoPNM1	000000 00000000 00000000 00000000 00000000 00000000 000 01111
	Output Data	111111 11111111 11111111 11111111 11111111 11111111 1 01 11111
	Match 011111	No
	Output truncated	111111 11111111 11111111 11111111 11111111 11111111 101

For dynamic and random rounding, the selection port for CMUX is opened to the user. When and how the two rounding patterns, RNDtoPN and RNDtoPNM1, are switched is up to the user.

9. sysDSP Slice Control Signal and Data Signal Descriptions

RST	Asynchronous reset of selected registers
SIGNEDA	Dynamic signal: 0 = unsigned, 1 = signed
SIGNEDB	Dynamic signal: 0 = unsigned, 1 = signed
ACCUMSLOAD	Dynamic signal: 0 = accumulate, 1 = load
ADDNSUB	Dynamic signal: 0 = subtract, 1 = add
SOURCEA	Dynamic signal: 0 = parallel input, 1 = shift input
SOURCEB	Dynamic signal: 0 = parallel input, 1 = shift input

Appendix A. DSP Primitives

```
module MULT18X18C (
A17,A16,A15,A14,A13,A12,A11,A10,A9,A8,A7,A6,A5,A4,A3,A2,A1,A0,
B17,B16,B15,B14,B13,B12,B11,B10,B9,B8,B7,B6,B5,B4,B3,B2,B1,B0,
SIGNEDA,SIGNEDB,SOURCEA,SOURCEB,
CE3,CE2,CE1,CE0,CLK3,CLK2,CLK1,CLK0,RST3,RST2,RST1,RST0,
SRIA17,SRIA16,SRIA15,SRIA14,SRIA13,SRIA12,SRIA11,SRIA10,SRIA9,SRIA8,SRIA7,SRIA6,
SRIA5,SRIA4,SRIA3,SRIA2,SRIA1,SRIA0,
SRIB17,SRIB16,SRIB15,SRIB14,SRIB13,SRIB12,SRIB11,SRIB10,SRIB9,SRIB8,SRIB7,SRIB6,
SRIB5,SRIB4,SRIB3,SRIB2,SRIB1,SRIB0,
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,ROA16,ROA15,ROA14,ROA13,ROA12,ROA11,ROA10,ROA9,ROA8,ROA7,ROA6,ROA5,ROA4,RO
A3,ROA2,ROA1,ROA0,
ROB17,ROB16,ROB15,ROB14,ROB13,ROB12,ROB11,ROB10,ROB9,ROB8,ROB7,ROB6,ROB5,ROB4,RO
B3,ROB2,ROB1,ROB0,
P35,P34,P33,P32,P31,P30,P29,P28,P27,P26,P25,P24,P23,P22,P21,P20,P19,
P18,P17,P16,P15,P14,P13,P12,P11,P10,P9,P8,P7,P6,P5,P4,P3,P2,P1,P0,SIGNEDP);
input A17,A16,A15,A14,A13,A12,A11,A10,A9,A8,A7,A6,A5,A4,A3,A2,A1,A0;
input B17,B16,B15,B14,B13,B12,B11,B10,B9,B8,B7,B6,B5,B4,B3,B2,B1,B0;
input SIGNEDA,SIGNEDB,SOURCEA,SOURCEB;
input CE3,CE2,CE1,CE0,CLK3,CLK2,CLK1,CLK0,RST3,RST2,RST1,RST0;
input SRIA17,SRIA16,SRIA15,SRIA14,SRIA13,SRIA12,SRIA11,SRIA10,SRIA9;
input SRIA8,SRIA7,SRIA6,SRIA5,SRIA4,SRIA3,SRIA2,SRIA1,SRIA0;
input SRIB17,SRIB16,SRIB15,SRIB14,SRIB13,SRIB12,SRIB11,SRIB10,SRIB9;
input SRIB8,SRIB7,SRIB6,SRIB5,SRIB4,SRIB3,SRIB2,SRIB1,SRIB0;
output SROA17,SROA16,SROA15,SROA14,SROA13,SROA12,SROA11,SROA10,SROA9;
output SROA8,SROA7,SROA6,SROA5,SROA4,SROA3,SROA2,SROA1,SROA0;
output SROB17,SROB16,SROB15,SROB14,SROB13,SROB12,SROB11,SROB10,SROB9;
output SROB8,SROB7,SROB6,SROB5,SROB4,SROB3,SROB2,SROB1,SROB0;
output
ROA17,ROA16,ROA15,ROA14,ROA13,ROA12,ROA11,ROA10,ROA9,ROA8,ROA7,ROA6,ROA5,ROA4,RO
A3,ROA2,ROA1,ROA0;
output
ROB17,ROB16,ROB15,ROB14,ROB13,ROB12,ROB11,ROB10,ROB9,ROB8,ROB7,ROB6,ROB5,ROB4,RO
B3,ROB2,ROB1,ROB0;
output P35,P34,P33,P32,P31,P30,P29,P28,P27,P26,P25,P24,P23,P22,P21,P20,P19,P18;
output P17,P16,P15,P14,P13,P12,P11,P10,P9,P8,P7,P6,P5,P4,P3,P2,P1,P0;
output SIGNEDP;
parameter REG_INPUTA_CLK = "NONE";
parameter REG_INPUTA_CE = "CE0";
parameter REG_INPUTA_RST = "RST0";
parameter REG_INPUTB_CLK = "NONE";
parameter REG_INPUTB_CE = "CE0";
parameter REG_INPUTB_RST = "RST0";
parameter REG_PIPELINE_CLK = "NONE";
parameter REG_PIPELINE_CE = "CE0";
parameter REG_PIPELINE_RST = "RST0";
parameter REG_OUTPUT_CLK = "NONE";
parameter REG_OUTPUT_CE = "CE0";
parameter REG_OUTPUT_RST = "RST0";
parameter CAS_MATCH_REG = "FALSE";
parameter MULT_BYPASS = "DISABLED";
parameter GSR = "ENABLED";
parameter RESETMODE = "SYNC";
endmodule
module MULT9X9C (
A8,A7,A6,A5,A4,A3,A2,A1,A0,B8,B7,B6,B5,B4,B3,B2,B1,B0,
SIGNEDA,SIGNEDB,SOURCEA,SOURCEB,
CE3,CE2,CE1,CE0,CLK3,CLK2,CLK1,CLK0,RST3,RST2,RST1,RST0,
SRIA8,SRIA7,SRIA6,SRIA5,SRIA4,SRIA3,SRIA2,SRIA1,SRIA0,
SRIB8,SRIB7,SRIB6,SRIB5,SRIB4,SRIB3,SRIB2,SRIB1,SRIB0,
SROA8,SROA7,SROA6,SROA5,SROA4,SROA3,SROA2,SROA1,SROA0,
SROB8,SROB7,SROB6,SROB5,SROB4,SROB3,SROB2,SROB1,SROB0,
ROA8,ROA7,ROA6,ROA5,ROA4,ROA3,ROA2,ROA1,ROA0,
ROB8,ROB7,ROB6,ROB5,ROB4,ROB3,ROB2,ROB1,ROB0,
P17,P16,P15,P14,P13,P12,P11,P10,P9,P8,P7,P6,P5,P4,P3,P2,P1,P0,
SIGNEDP,
);
input A8,A7,A6,A5,A4,A3,A2,A1,A0;
input B8,B7,B6,B5,B4,B3,B2,B1,B0;
input SIGNEDA,SIGNEDB,SOURCEA,SOURCEB;
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3;
input SRIA8,SRIA7,SRIA6,SRIA5,SRIA4,SRIA3,SRIA2,SRIA1,SRIA0;
input SRIB8,SRIB7,SRIB6,SRIB5,SRIB4,SRIB3,SRIB2,SRIB1,SRIB0;
output SROA8,SROA7,SROA6,SROA5,SROA4,SROA3,SROA2,SROA1,SROA0;
output SROB8,SROB7,SROB6,SROB5,SROB4,SROB3,SROB2,SROB1,SROB0;
output ROA8,ROA7,ROA6,ROA5,ROA4,ROA3,ROA2,ROA1,ROA0;
output ROB8,ROB7,ROB6,ROB5,ROB4,ROB3,ROB2,ROB1,ROB0;
output P17,P16,P15,P14,P13,P12,P11,P10,P9,P8,P7,P6,P5,P4,P3,P2,P1,P0;
output SIGNEDP;
parameter REG_INPUTA_CLK = "NONE";
parameter REG_INPUTA_CE = "CE0";
parameter REG_INPUTA_RST = "RST0";
parameter REG_INPUTB_CLK = "NONE";
parameter REG_INPUTB_CE = "CE0";
parameter REG_INPUTB_RST = "RST0";
parameter REG_PIPELINE_CLK = "NONE";
parameter REG_PIPELINE_CE = "CE0";
parameter REG_PIPELINE_RST = "RST0";
parameter REG_OUTPUT_CLK = "NONE";
parameter REG_OUTPUT_CE = "CE0";
parameter REG_OUTPUT_RST = "RST0";
parameter CAS_MATCH_REG = "NONE";
parameter MULT_BYPASS = "DISABLED";
parameter GSR = "ENABLED";
parameter RESETMODE = "SYNC";
endmodule
module ALU54A(
CE3,CE2,CE1,CE0,CLK3,CLK2,CLK1,CLK0,RST3,RST2,RST1,RST0,SIGNEDIA,SIGNEDIB,
A35,A34,A33,A32,A31,A30,A29,A28,A27,A26,A25,A24,A23,A22,A21,A20,A19,A18,
A17,A16,A15,A14,A13,A12,A11,A10,A9,A8,A7,A6,A5,A4,A3,A2,A1,A0,
B35,B34,B33,B32,B31,B30,B29,B28,B27,B26,B25,B24,B23,B22,B21,B20,B19,B18,
B17,B16,B15,B14,B13,B12,B11,B10,B9,B8,B7,B6,B5,B4,B3,B2,B1,B0,
C53,C52,C51,C50,C49,C48,C47,C46,C45,C44,C43,C42,C41,C40,C39,C38,C37,C36,
C35,C34,C33,C32,C31,C30,C29,C28,C27,C26,C25,C24,C23,C22,C21,C20,C19,C18,
C17,C16,C15,C14,C13,C12,C11,C10,C9,C8,C7,C6,C5,C4,C3,C2,C1,C0,
MA35,MA34,MA33,MA32,MA31,MA30,MA29,MA28,MA27,MA26,MA25,MA24,MA23,MA22,MA21,
MA20,MA19,MA18,MA17,MA16,MA15,MA14,MA13,MA12,MA11,MA10,MA9,MA8,MA7,MA6,MA5,
MA4,MA3,MA2,MA1,MA0,
MB35,MB34,MB33,MB32,MB31,MB30,MB29,MB28,MB27,MB26,MB25,MB24,MB23,MB22,MB21,
MB20,MB19,MB18,MB17,MB16,MB15,MB14,MB13,MB12,MB11,MB10,MB9,MB8,MB7,MB6,MB5,
MB4,MB3,MB2,MB1,MB0,
CIN53,CIN52,CIN51,CIN50,CIN49,CIN48,CIN47,CIN46,CIN45,CIN44,CIN43,CIN42,
CIN41,CIN40,CIN39,CIN38,CIN37,CIN36,CIN35,CIN34,CIN33,CIN32,CIN31,CIN30,CIN29,
CIN28,CIN27,CIN26,CIN25,CIN24,CIN23,CIN22,CIN21,CIN20,CIN19,CIN18,CIN17,CIN16,
CIN15,CIN14,CIN13,CIN12,CIN11,CIN10,CIN9,CIN8,CIN7,CIN6,CIN5,CIN4,CIN3,CIN2,CIN1
,CIN0,
OP10,OP9,OP8,OP7,OP6,OP5,OP4,OP3,OP2,OP1,OP0,SIGNEDCIN,
R53,R52,R51,R50,R49,R48,R47,R46,R45,R44,R43,R42,R41,R40,R39,R38,R37,R36,
R35,R34,R33,R32,R31,R30,R29,R28,R27,R26,R25,R24,R23,R22,R21,R20,R19,R18,
R17,R16,R15,R14,R13,R12,R11,R10,R9,R8,R7,R6,R5,R4,R3,R2,R1,R0,
EQZ,EQZM,EQOM,EQPAT,EQPATB,OVER,UNDER,OVERUNDER,SIGNEDR);
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3,SIGNEDIA,SIGNEDIB;
input A35,A34,A33,A32,A31,A30,A29,A28,A27,A26,A25,A24,A23,A22,A21,A20,A19,A18;
input A17,A16,A15,A14,A13,A12,A11,A10,A9,A8,A7,A6,A5,A4,A3,A2,A1,A0;
input B35,B34,B33,B32,B31,B30,B29,B28,B27,B26,B25,B24,B23,B22,B21,B20,B19,B18;
input B17,B16,B15,B14,B13,B12,B11,B10,B9,B8,B7,B6,B5,B4,B3,B2,B1,B0;
input C53,C52,C51,C50,C49,C48,C47,C46,C45,C44,C43,C42,C41,C40,C39,C38,C37,C36;
input C35,C34,C33,C32,C31,C30,C29,C28,C27,C26,C25,C24,C23,C22,C21,C20,C19,C18;
input C17,C16,C15,C14,C13,C12,C11,C10,C9,C8,C7,C6,C5,C4,C3,C2,C1,C0;
input
MA35,MA34,MA33,MA32,MA31,MA30,MA29,MA28,MA27,MA26,MA25,MA24,MA23,MA22,MA21;
input
MA20,MA19,MA18,MA17,MA16,MA15,MA14,MA13,MA12,MA11,MA10,MA9,MA8,MA7,MA6,MA5;
input MA4,MA3,MA2,MA1,MA0;
input
MB35,MB34,MB33,MB32,MB31,MB30,MB29,MB28,MB27,MB26,MB25,MB24,MB23,MB22,MB21;
input
MB20,MB19,MB18,MB17,MB16,MB15,MB14,MB13,MB12,MB11,MB10,MB9,MB8,MB7,MB6,MB5;
input MB4,MB3,MB2,MB1,MB0;
input CIN53,CIN52,CIN51,CIN50,CIN49,CIN48,CIN47,CIN46,CIN45,CIN44,CIN43,CIN42;
input
CIN41,CIN40,CIN39,CIN38,CIN37,CIN36,CIN35,CIN34,CIN33,CIN32,CIN31,CIN30,CIN29;
input
CIN28,CIN27,CIN26,CIN25,CIN24,CIN23,CIN22,CIN21,CIN20,CIN19,CIN18,CIN17,CIN16;
input
CIN15,CIN14,CIN13,CIN12,CIN11,CIN10,CIN9,CIN8,CIN7,CIN6,CIN5,CIN4,CIN3,CIN2,CIN1
,CIN0;
input OP10,OP9,OP8,OP7,OP6,OP5,OP4,OP3,OP2,OP1,OP0,SIGNEDCIN;
output R53,R52,R51,R50,R49,R48,R47,R46,R45,R44,R43,R42,R41,R40,R39,R38,R37,R36;
output R35,R34,R33,R32,R31,R30,R29,R28,R27,R26,R25,R24,R23,R22,R21,R20,R19,R18;
output R17,R16,R15,R14,R13,R12,R11,R10,R9,R8,R7,R6,R5,R4,R3,R2,R1,R0;
output EQZ,EQZM,EQOM,EQPAT,EQPATB,OVER,UNDER,OVERUNDER,SIGNEDR;
parameter REG_INPUTC0_CLK = "NONE";
parameter REG_INPUTC0_CE = "CE0";
parameter REG_INPUTC0_RST = "RST0";
parameter REG_INPUTC1_CLK = "NONE";
parameter REG_INPUTC1_CE = "CE0";
parameter REG_INPUTC1_RST = "RST0";
parameter REG_OPCODEOP0_0_CLK = "NONE";
parameter REG_OPCODEOP0_0_CE = "CE0";
parameter REG_OPCODEOP0_0_RST = "RST0";
parameter REG_OPCODEOP1_0_CLK = "NONE";
parameter REG_OPCODEOP0_1_CLK = "NONE";
parameter REG_OPCODEOP0_1_CE = "CE0";
parameter REG_OPCODEOP0_1_RST = "RST0";
parameter REG_OPCODEOP1_1_CLK = "NONE";
parameter REG_OPCODEIN_0_CLK = "NONE";
parameter REG_OPCODEIN_0_CE = "CE0";
parameter REG_OPCODEIN_0_RST = "RST0";
parameter REG_OPCODEIN_1_CLK = "NONE";
parameter REG_OPCODEIN_1_CE = "CE0";
parameter REG_OPCODEIN_1_RST = "RST0";
parameter REG_OUTPUT0_CLK = "NONE";
parameter REG_OUTPUT0_CE = "CE0";
parameter REG_OUTPUT0_RST = "RST0";
parameter REG_OUTPUT1_CLK = "NONE";
parameter REG_OUTPUT1_CE = "CE0";
parameter REG_OUTPUT1_RST = "RST0";
parameter REG_FLAG_CLK = "NONE";
parameter REG_FLAG_CE = "CE0";
parameter REG_FLAG_RST = "RST0";
parameter MCPAT_SOURCE = "STATIC";
parameter MASKPAT_SOURCE = "STATIC";
parameter MASK01 = "0x00000000000000";
parameter MCPAT = "0x00000000000000";
parameter MASKPAT = "0x00000000000000";
parameter RNDPAT = "0x00000000000000";
parameter GSR = "ENABLED";
parameter RESETMODE = "SYNC";
parameter MULT9_MODE = "DISABLED";
parameter FORCE_ZERO_BARREL_SHIFT = "DISABLED";
parameter LEGACY = "DISABLED";
endmodule
module ALU24A(
MA17,MA16,MA15,MA14,MA13,MA12,MA11,MA10,MA9,MA8,MA7,MA6,MA5,
MA4,MA3,MA2,MA1,MA0,
MB17,MB16,MB15,MB14,MB13,MB12,MB11,MB10,MB9,MB8,MB7,MB6,MB5,
MB4,MB3,MB2,MB1,MB0,
CIN23,CIN22,CIN21,CIN20,CIN19,CIN18,CIN17,CIN16,
CIN15,CIN14,CIN13,CIN12,CIN11,CIN10,CIN9,CIN8,CIN7,CIN6,CIN5,CIN4,CIN3,CIN2,CIN1
,CIN0,
CE3,CE2,CE1,CE0,CLK3,CLK2,CLK1,CLK0,RST3,RST2,RST1,RST0,SIGNEDIA,SIGNEDIB,OPADDN
SUB,OPCINSEL,
R23,R22,R21,R20,R19,R18,
R17,R16,R15,R14,R13,R12,R11,R10,R9,R8,R7,R6,R5,R4,R3,R2,R1,R0);
input MA17,MA16,MA15,MA14,MA13,MA12,MA11,MA10,MA9,MA8,MA7,MA6,MA5;
input MA4,MA3,MA2,MA1,MA0;
input MB17,MB16,MB15,MB14,MB13,MB12,MB11,MB10,MB9,MB8,MB7,MB6,MB5;
input MB4,MB3,MB2,MB1,MB0;
input CIN23,CIN22,CIN21,CIN20,CIN19,CIN18,CIN17,CIN16;
```

```
input
CIN15,CIN14,CIN13,CIN12,CIN11,CIN10,CIN9,CIN8,CIN7,CIN6,CIN5,CIN4,CIN3,CIN2,CIN1
,CIN0;
input
CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3,SIGNEDIA,SIGNEDIB,OPADDN
SUB,OPCINSEL;
output R23,R22,R21,R20,R19,R18;
output R17,R16,R15,R14,R13,R12,R11,R10,R9,R8,R7,R6,R5,R4,R3,R2,R1,R0;
parameter REG_OUTPUT_CLK = "NONE";
parameter REG_OUTPUT_CE = "CE0";
parameter REG_OUTPUT_RST = "RST0";
parameter REG_OPCODE_0_CLK = "NONE";
parameter REG_OPCODE_0_CE = "CE0";
parameter REG_OPCODE_0_RST = "RST0";
parameter REG_OPCODE_1_CLK = "NONE";
parameter REG_OPCODE_1_CE = "CE0";
parameter REG_OPCODE_1_RST = "RST0";
parameter GSR = "ENABLED";
parameter RESETMODE = "SYNC";
endmodule
```

Appendix B. Using IPexpress for Diamond

B.1. Invoking IPexpress for Diamond

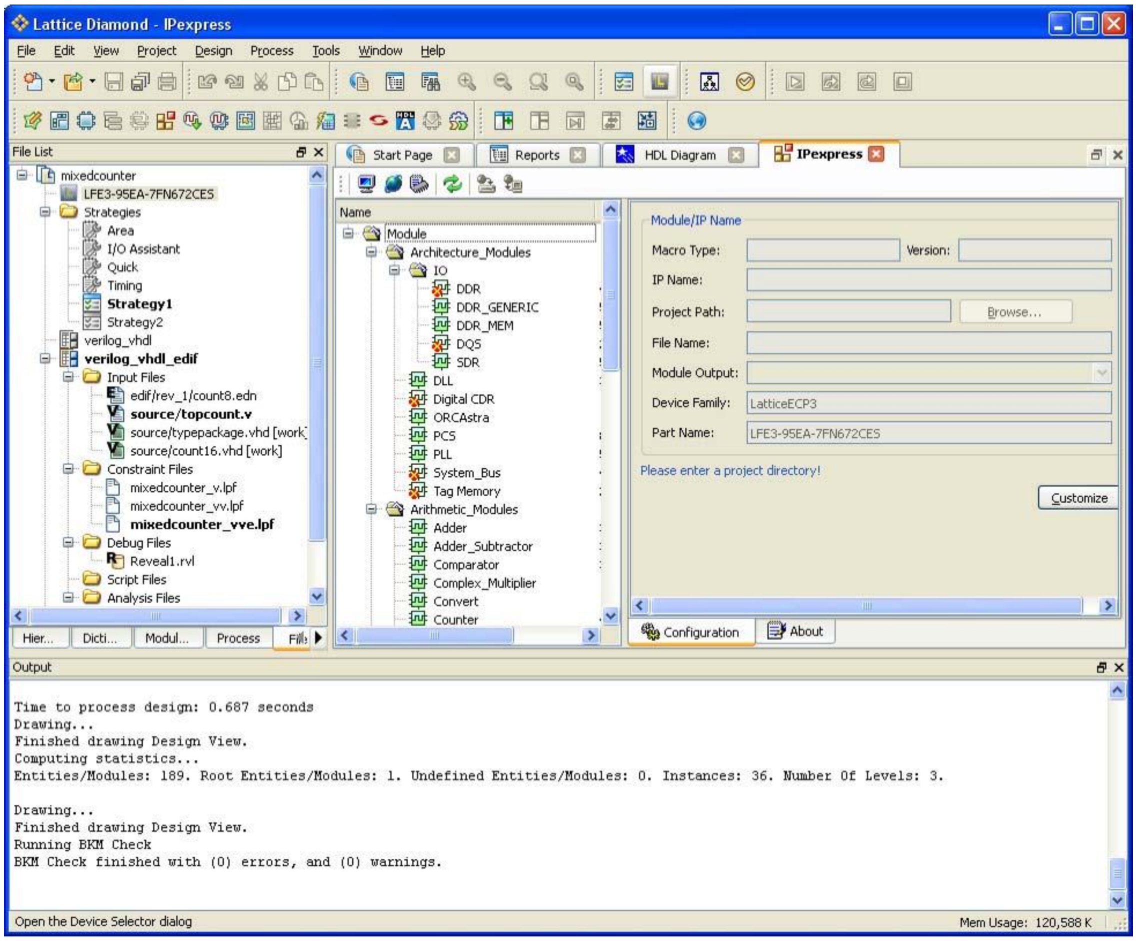
There are several ways IPexpress can be invoked. To invoke IPexpress from the Start menu, select:

Start > Programs > Lattice Diamond 1.0 > Accessories > IPexpress

To invoke IPexpress from within Diamond, a project must be opened, then invoke the IPexpress icon or select:

Tools > IPexpress

The IPexpress interface appears as shown below:



```
mixedcounter
  LFE3-95EA-7FN672CES
  Strategies
    Area
    I/O Assistant
    Quick
    Timing
    Strategy1
    Strategy2
  verilog_vhdl
  verilog_vhdl_edif
    Input Files
      edif/rev_1/count8.edn
      source/topcount.v
      source/typepackage.vhd [work]
      source/count16.vhd [work]
    Constraint Files
      mixedcounter_v.lpf
      mixedcounter_vv.lpf
      mixedcounter_vve.lpf
    Debug Files
      Reveal1.rvl
    Script Files
    Analysis Files
```

```
Module
  Architecture_Modules
    IO
      DDR
      DDR_GENERIC
      DDR_MEM
      DQS
      SDR
    DLL
    Digital CDR
    ORCAstra
    PCS
    PLL
    System_Bus
    Tag Memory
  Arithmetic_Modules
    Adder
    Adder_Subtractor
    Comparator
    Complex_Multiplier
    Convert
    Counter
```

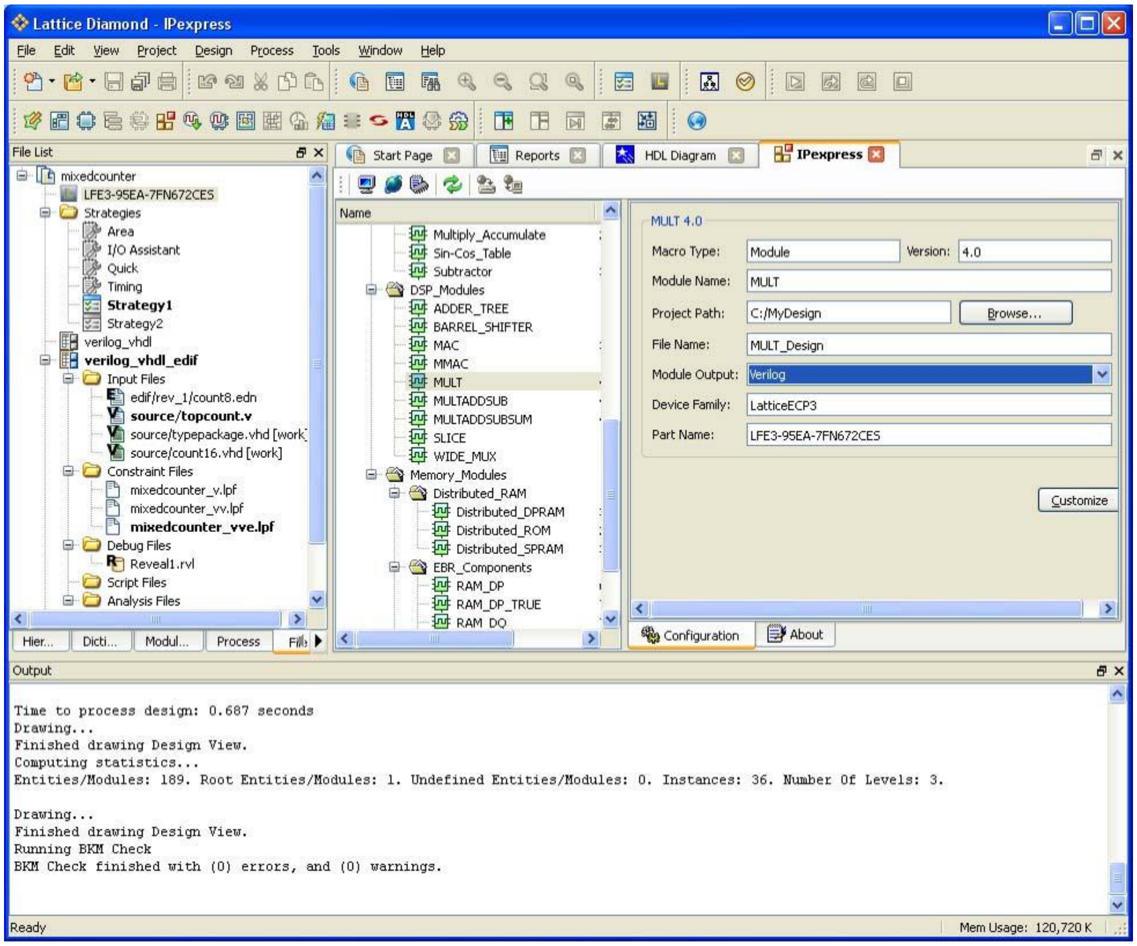
Module/IP Name
Macro Type: Version:
IP Name:
Project Path:
File Name:
Module Output:
Device Family: LatticeECP3
Part Name: LFE3-95EA-7FN672CES
Please enter a project directory!

```
Time to process design: 0.687 seconds
Drawing...
Finished drawing Design View.
Computing statistics...
Entities/Modules: 189. Root Entities/Modules: 1. Undefined Entities/Modules: 0. Instances: 36. Number Of Levels: 3.

Drawing...
Finished drawing Design View.
Running BKM Check
BKM Check finished with (0) errors, and (0) warnings.
```

Open the Device Selector dialog   Mem Usage: 120,588 K

Figure B.1. IPexpress Interface

Scroll to the modules and left-click on the module you wish to use. Enter the information into the IPexpress interface as shown in the example Figure B.2.



MULT 4.0

Macro Type: Module
Version: 4.0
Module Name: MULT
Project Path: C:/MyDesign
File Name: MULT_Design
Module Output: Verilog
Device Family: LatticeECP3
Part Name: LFE3-95EA-7FN672CES

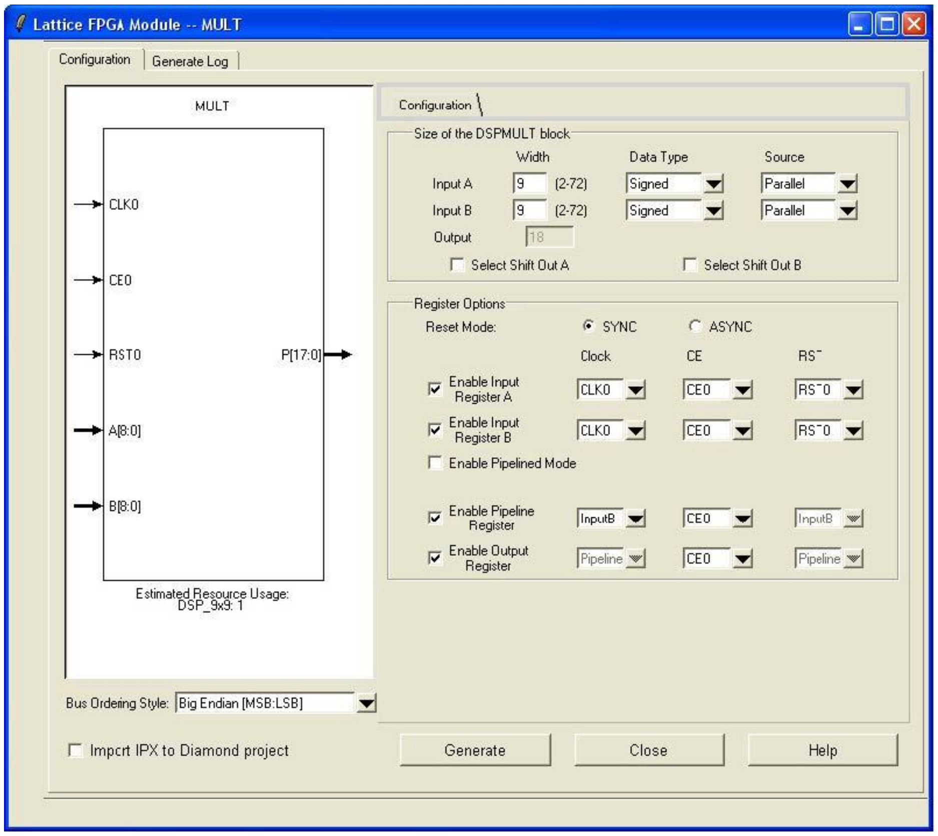
```
Time to process design: 0.687 seconds
Drawing...
Finished drawing Design View.
Computing statistics...
Entities/Modules: 189. Root Entities/Modules: 1. Undefined Entities/Modules: 0. Instances: 36. Number Of Levels: 3.

Drawing...
Finished drawing Design View.
Running BKM Check
BKM Check finished with (0) errors, and (0) warnings.
```

Ready   Mem Usage: 120,720 K

Figure B.2. Creating the Module Instance

Select **Customize**.

An example of a MULT Module Dialog window appears as shown in Figure B.3. Select **Help** for information about the fields in this window.



Estimated Resource Usage:
DSP_9x9: 1

Bus Ordering Style: Big Endian [MSB:LSB]

☐ Import IPX to Diamond project

Configuration

Size of the DSPMULT block

	Width		Data Type	Source
Input A	9	(2-72)	Signed	Parallel
Input B	9	(2-72)	Signed	Parallel
Output	18			

☐ Select Shift Out A ☐ Select Shift Out B

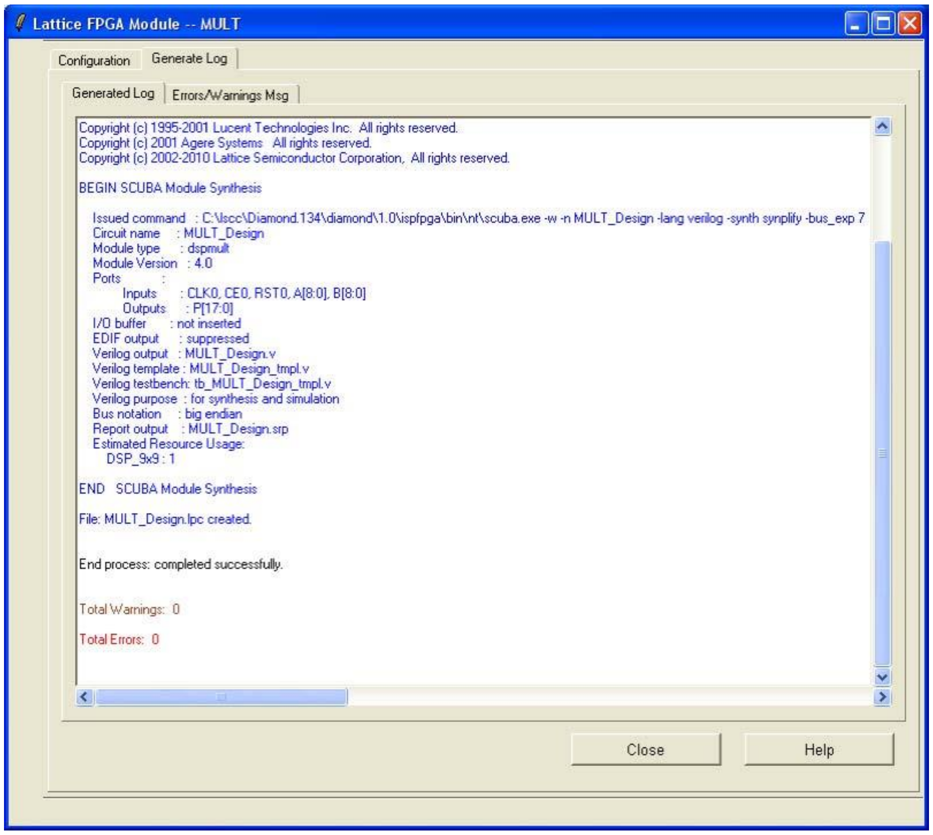
Register Options

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	Pipeline	CE0	Pipeline

Figure B.3. MULT Module

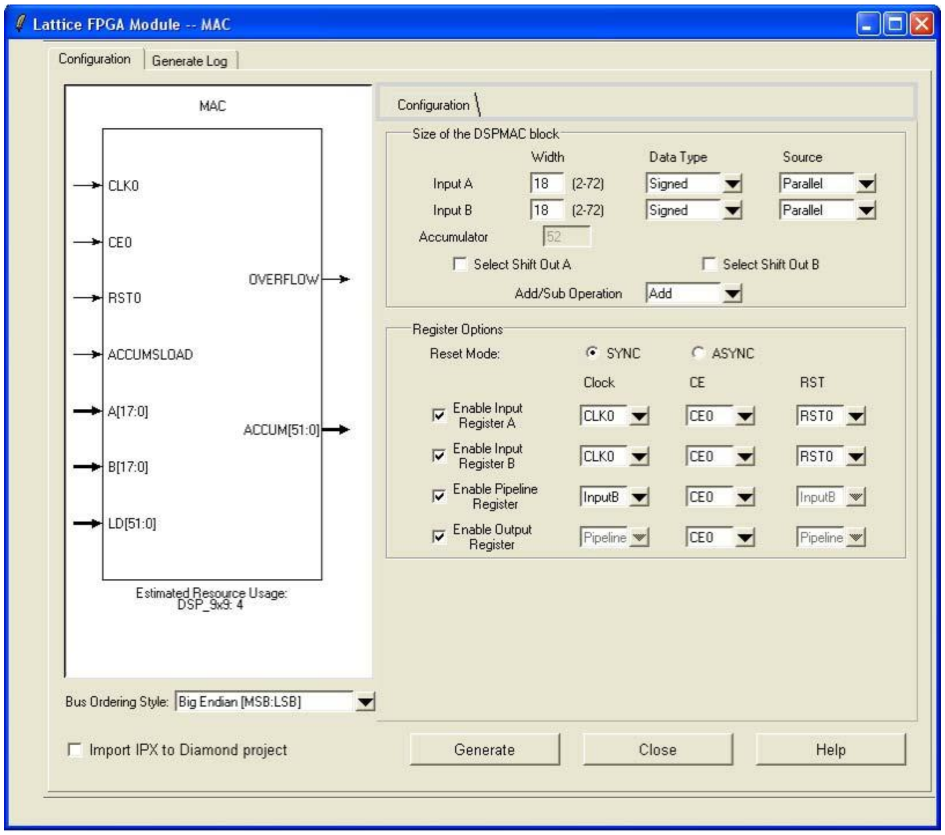
When you are finished selecting your options, Select **Generate**. A log window will appear similar to what is shown in Figure B.4.



```
Copyright (c) 1995-2001 Lucent Technologies Inc.  All rights reserved.
Copyright (c) 2001 Agere Systems   All rights reserved.
Copyright (c) 2002-2010 Lattice Semiconductor Corporation,  All rights reserved.

BEGIN SCUBA Module Synthesis

    Issued command   : C:\lscc\Diamond.134\diamond\1.0\ispfpga\bin\nt\scuba.exe -w -n MULT_Design -lang verilog -synth synplify -bus_exp 7
    Circuit name     : MULT_Design
    Module type      : dspmult
    Module Version   : 4.0
    Ports            :
        Inputs       : CLK0, CE0, RST0, A[8:0], B[8:0]
        Outputs      : P[17:0]
    I/O buffer       : not inserted
    EDIF output      : suppressed
    Verilog output   : MULT_Design.v
    Verilog template : MULT_Design_tmpl.v
    Verilog testbench: tb_MULT_Design_tmpl.v
    Verilog purpose  : for synthesis and simulation
    Bus notation     : big endian
    Report output    : MULT_Design.srp
    Estimated Resource Usage:
        DSP_9x9 : 1

END   SCUBA Module Synthesis

File: MULT_Design.lpc created.


End process: completed successfully.


Total Warnings:  0

Total Errors:  0
```

Figure B.4. IP Generation Log Window

All other DSP Module interfaces for the LatticeECP3 are shown in the figures below.



Lattice FPGA Module -- MAC

Configuration | Generate Log

MAC

CLK0, CE0, RST0, ACCUMSLOAD, A[17:0], B[17:0], LD[51:0] → OVERFLOW, ACCUM[51:0]

Estimated Resource Usage:
DSP_9x9: 4

Bus Ordering Style: Big Endian [MSB:LSB]

Import IPX to Diamond project: ☐

Configuration

Size of the DSPMAC block

	Width		Data Type	Source
Input A	18	(2-72)	Signed	Parallel
Input B	18	(2-72)	Signed	Parallel
Accumulator	52			

☐ Select Shift Out A ☐ Select Shift Out B

Add/Sub Operation: Add

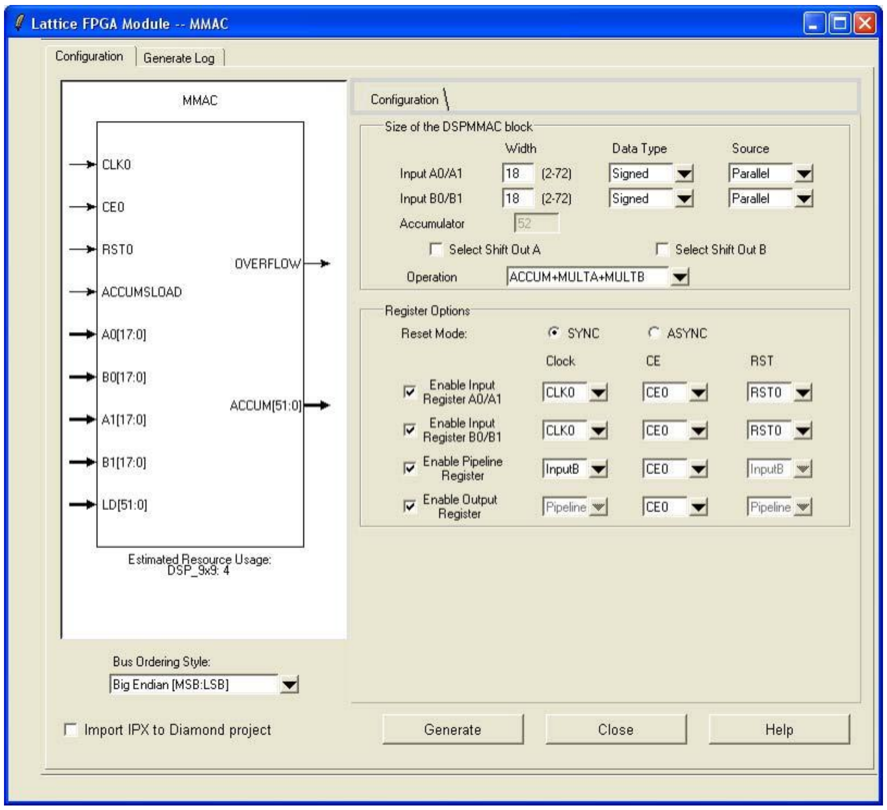
Register Options

Reset Mode: [✓ SYNC] ASYNC

	Clock	CE	RST
☑ Enable Input Register A	CLK0	CE0	RST0
☑ Enable Input Register B	CLK0	CE0	RST0
☑ Enable Pipeline Register	InputB	CE0	InputB
☑ Enable Output Register	Pipeline	CE0	Pipeline

Generate | Close | Help

Figure B.5. MAC Module



Lattice FPGA Module -- MMAC

Configuration | Generate Log

MMAC

CLK0, CE0, RST0, ACCUMSLOAD, A0[17:0], B0[17:0], A1[17:0], B1[17:0], LD[51:0] → OVERFLOW, ACCUM[51:0]

Estimated Resource Usage:
DSP_9x9: 4

Bus Ordering Style: Big Endian [MSB:LSB]

Import IPX to Diamond project: ☐

Configuration

Size of the DSPMMAC block

	Width		Data Type	Source
Input A0/A1	18	(2-72)	Signed	Parallel
Input B0/B1	18	(2-72)	Signed	Parallel
Accumulator	52			

☐ Select Shift Out A ☐ Select Shift Out B

Operation: ACCUM+MULTA+MULTB

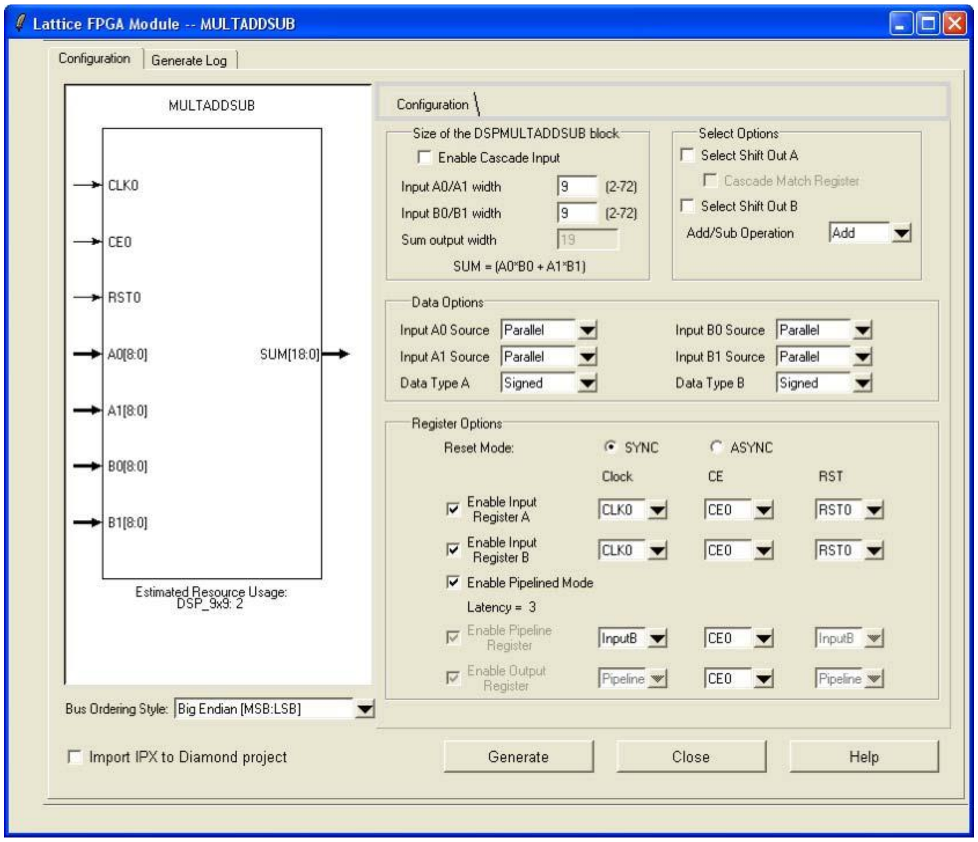
Register Options

Reset Mode: [✓ SYNC] ASYNC

	Clock	CE	RST
☑ Enable Input Register A0/A1	CLK0	CE0	RST0
☑ Enable Input Register B0/B1	CLK0	CE0	RST0
☑ Enable Pipeline Register	InputB	CE0	InputB
☑ Enable Output Register	Pipeline	CE0	Pipeline

Generate | Close | Help

Figure B.6. MMAC Module



Estimated Resource Usage:
DSP_9x9: 2

Bus Ordering Style: Big Endian [MSB:LSB]

☐ Import IPX to Diamond project

Configuration

Size of the DSPMULTADDSUB block
☐ Enable Cascade Input
Input A0/A1 width: 9 (2-72)
Input B0/B1 width: 9 (2-72)
Sum output width: 19
SUM = (A0*B0 + A1*B1)

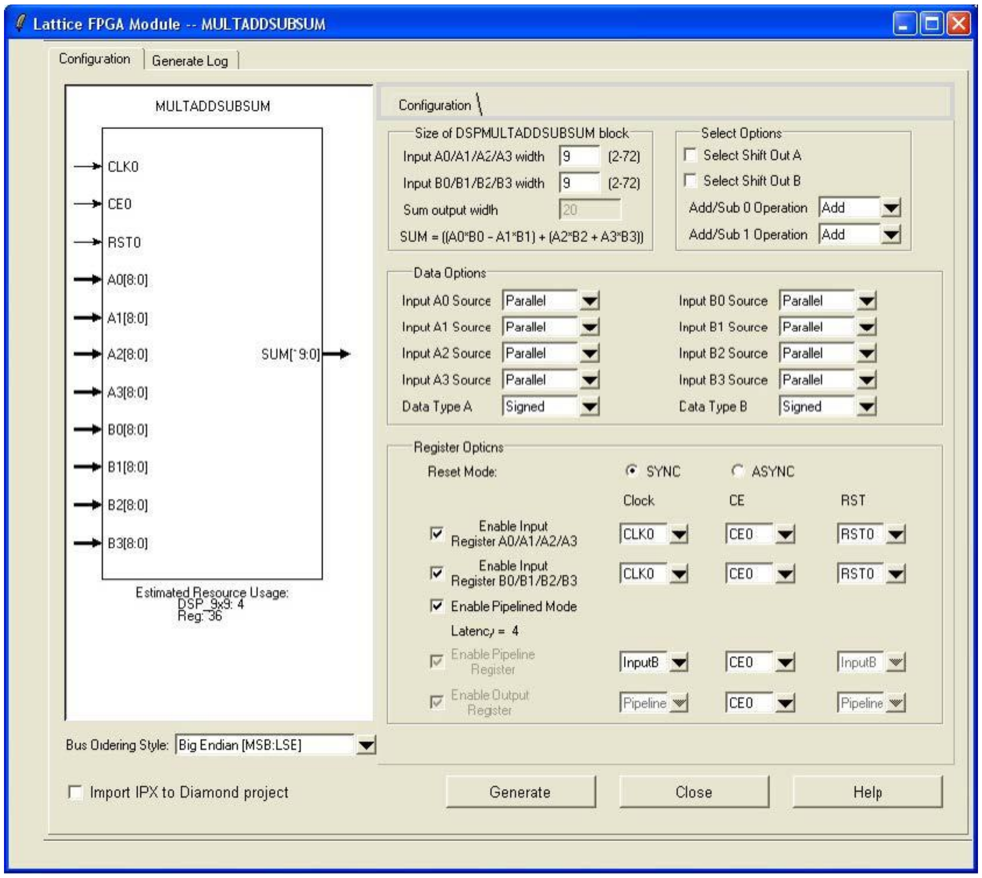
Select Options
☐ Select Shift Out A
☐ Cascade Match Register
☐ Select Shift Out B
Add/Sub Operation: Add

Data Options
Input A0 Source: Parallel
Input A1 Source: Parallel
Data Type A: Signed
Input B0 Source: Parallel
Input B1 Source: Parallel
Data Type B: Signed

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			
Latency = 3			
☑ Enable Pipeline Register	InputB	CE0	InputB
☑ Enable Output Register	Pipeline	CE0	Pipeline

Figure B.7. MULTADDSUB Module



Estimated Resource Usage:
DSP_9x9: 4
Reg: 36

Size of DSPMULTADDSUBSUM block
Input A0/A1/A2/A3 width: 9 (2-72)
Input B0/B1/B2/B3 width: 9 (2-72)
Sum output width: 20
SUM = ((A0*B0 - A1*B1) + (A2*B2 + A3*B3))

Select Options
☐ Select Shift Out A
☐ Select Shift Out B
Add/Sub 0 Operation: Add
Add/Sub 1 Operation: Add

Data Options

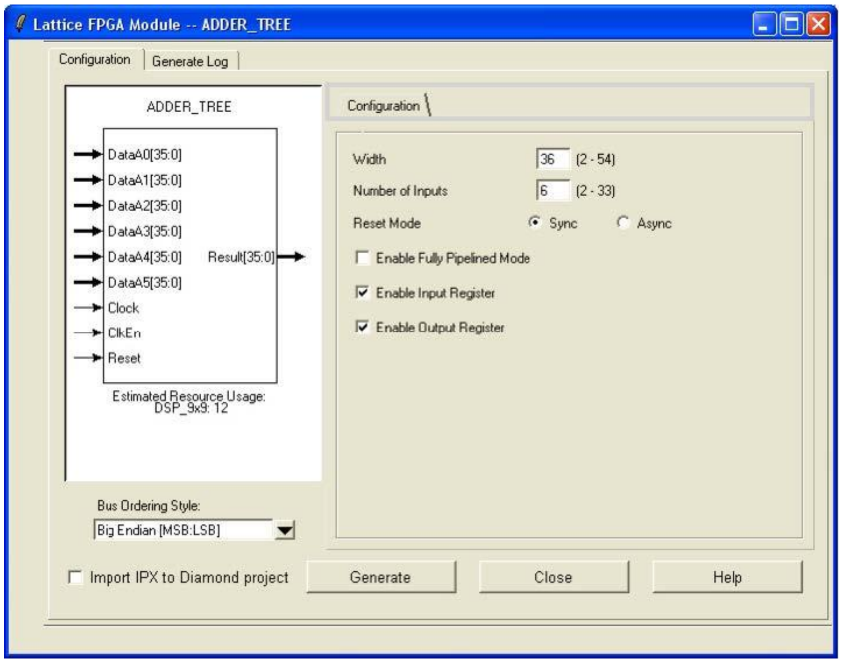
Input A0 Source	Parallel	Input B0 Source	Parallel
Input A1 Source	Parallel	Input B1 Source	Parallel
Input A2 Source	Parallel	Input B2 Source	Parallel
Input A3 Source	Parallel	Input B3 Source	Parallel
Data Type A	Signed	Data Type B	Signed

Register Options
Reset Mode: ● SYNC ○ ASYNC

	Clock	CE	RST
☑ Enable Input Register A0/A1/A2/A3	CLK0	CE0	RST0
☑ Enable Input Register B0/B1/B2/B3	CLK0	CE0	RST0
☑ Enable Pipelined Mode			
Latency = 4			
☑ Enable Pipeline Register	InputB	CE0	InputB
☑ Enable Output Register	Pipeline	CE0	Pipeline

Bus Ordering Style: Big Endian [MSB:LSB]
☐ Import IPX to Diamond project

Figure B.8. MULTADDSUBSUM Module

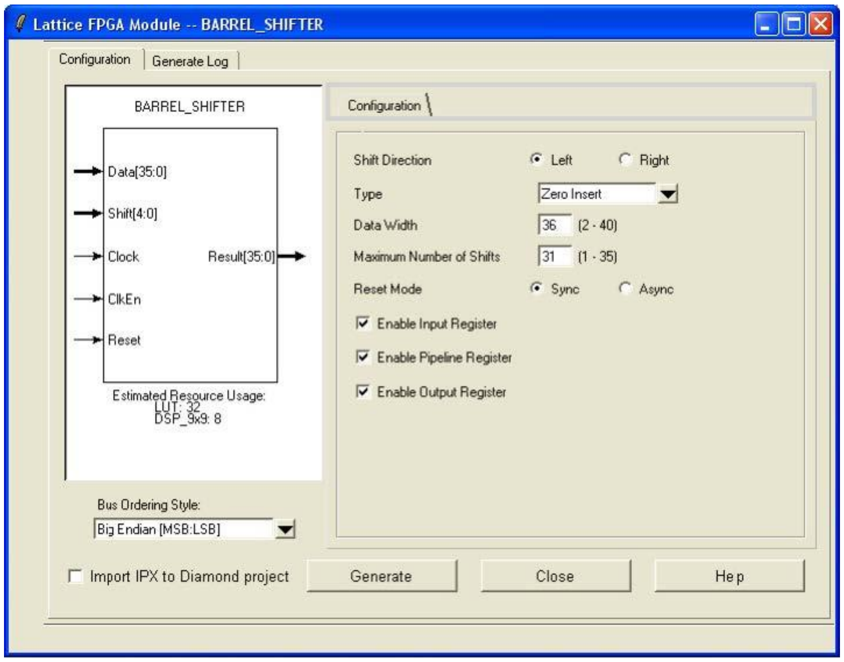


Estimated Resource Usage:
DSP_9x9: 12

Width: 36 (2 - 54)
Number of Inputs: 6 (2 - 33)
Reset Mode: ● Sync ○ Async
☐ Enable Fully Pipelined Mode
☑ Enable Input Register
☑ Enable Output Register

Bus Ordering Style: Big Endian [MSB:LSB]
☐ Import IPX to Diamond project

Figure B.9. ADDER_TREE Module

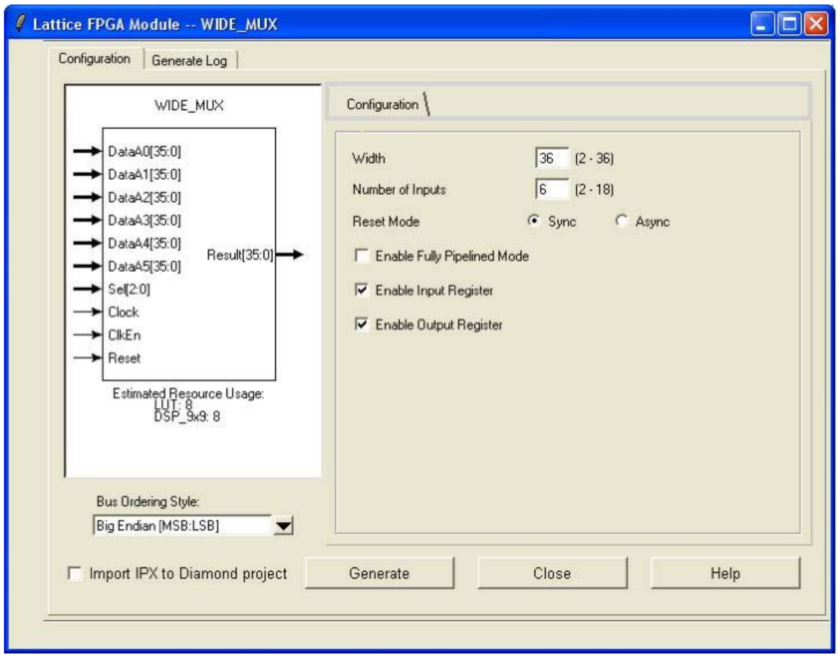


Estimated Resource Usage:
LUT: 32
DSP_9x9: 8

Bus Ordering Style: Big Endian [MSB:LSB]

Shift Direction: [✓ Left] Right
Type: Zero Insert
Data Width: 36 (2 - 40)
Maximum Number of Shifts: 31 (1 - 35)
Reset Mode: [✓ Sync] Async
☑ Enable Input Register
☑ Enable Pipeline Register
☑ Enable Output Register
☐ Import IPX to Diamond project

Figure B.10. BARREL_SHIFTER Module

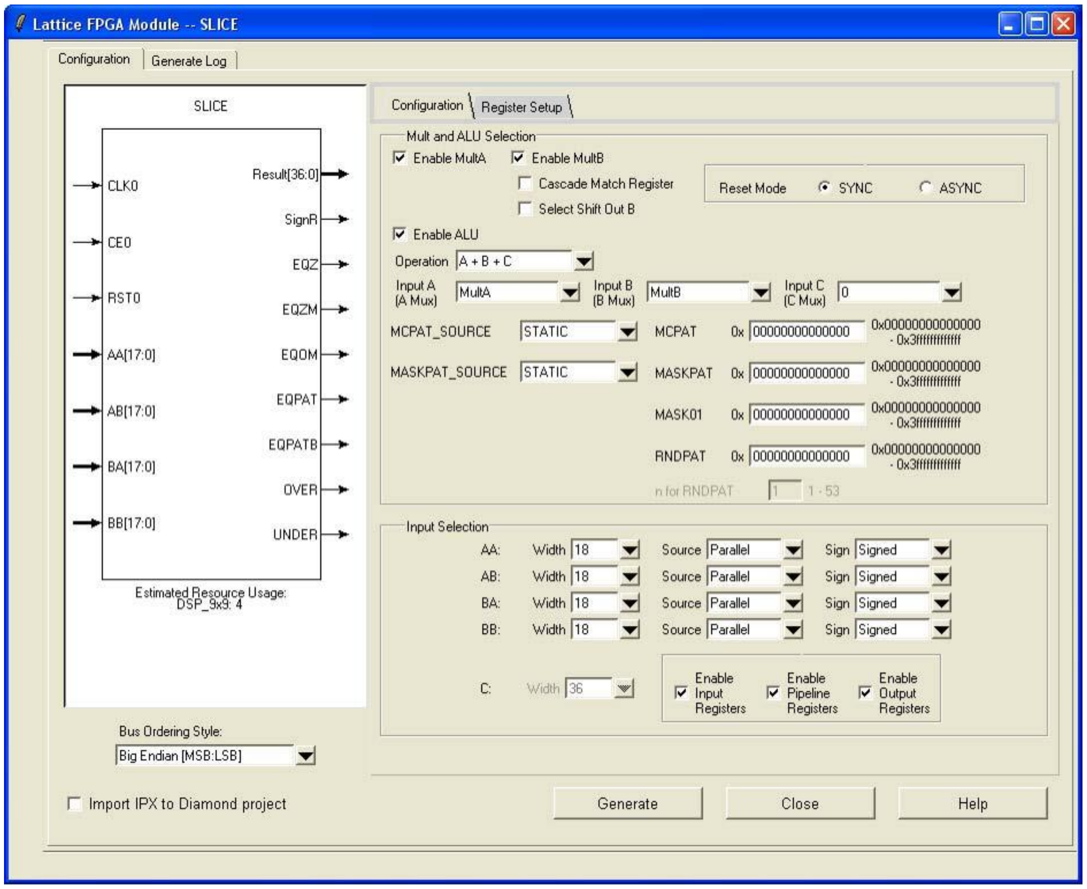


Estimated Resource Usage:
LUT: 8
DSP_9x9: 8

Bus Ordering Style: Big Endian [MSB:LSB]

Width: 36 (2 - 36)
Number of Inputs: 6 (2 - 18)
Reset Mode: [✓ Sync] Async
☐ Enable Fully Pipelined Mode
☑ Enable Input Register
☑ Enable Output Register
☐ Import IPX to Diamond project

Figure B.11. WIDE_MAX Module



Estimated Resource Usage:
DSP_9x9: 4

Bus Ordering Style: Big Endian [MSB:LSB]

☐ Import IPX to Diamond project

Mult and ALU Selection

☑ Enable MultA ☑ Enable MultB
☐ Cascade Match Register
☐ Select Shift Out B

Reset Mode: (● SYNC) ○ ASYNC

☑ Enable ALU

Operation: A + B + C

Input A (A Mux): MultA; Input B (B Mux): MultB; Input C (C Mux): 0

MCPAT_SOURCE: STATIC; MCPAT 0x 00000000000000 (0x00000000000000 - 0x3fffffffffffff)

MASKPAT_SOURCE: STATIC; MASKPAT 0x 00000000000000 (0x00000000000000 - 0x3fffffffffffff)

MASK01 0x 00000000000000 (0x00000000000000 - 0x3fffffffffffff)

RNDPAT 0x 00000000000000 (0x00000000000000 - 0x3fffffffffffff)

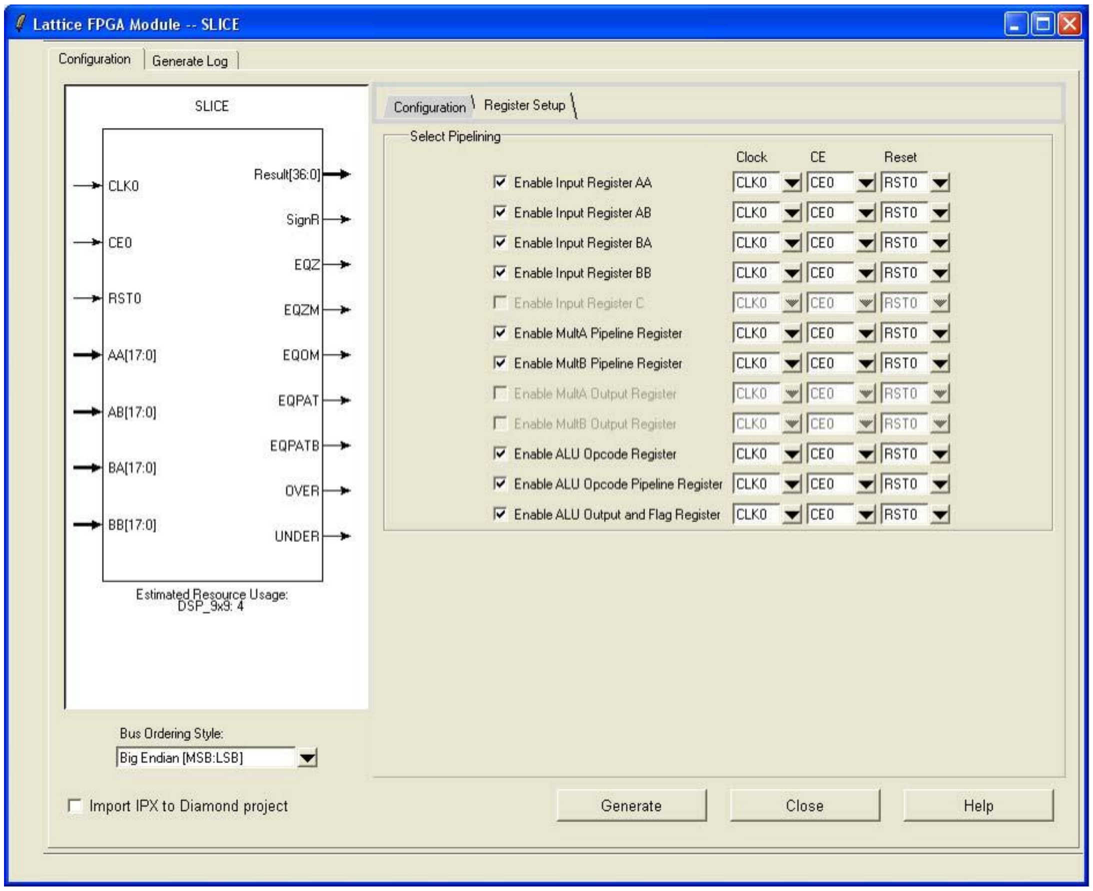
n for RNDPAT: 1 (1 - 53)

Input Selection

AA:	Width 18	Source Parallel	Sign Signed
AB:	Width 18	Source Parallel	Sign Signed
BA:	Width 18	Source Parallel	Sign Signed
BB:	Width 18	Source Parallel	Sign Signed
C:	Width 36		

☑ Enable Input Registers ☑ Enable Pipeline Registers ☑ Enable Output Registers

Figure B.12. SLICE Module – Configuration



	Clock	CE	Reset
[✓] Enable Input Register AA	CLK0	CE0	RST0
[✓] Enable Input Register AB	CLK0	CE0	RST0
[✓] Enable Input Register BA	CLK0	CE0	RST0
[✓] Enable Input Register BB	CLK0	CE0	RST0
[ ] Enable Input Register C	CLK0	CE0	RST0
[✓] Enable MultA Pipeline Register	CLK0	CE0	RST0
[✓] Enable MultB Pipeline Register	CLK0	CE0	RST0
[ ] Enable MultA Output Register	CLK0	CE0	RST0
[ ] Enable MultB Output Register	CLK0	CE0	RST0
[✓] Enable ALU Opcode Register	CLK0	CE0	RST0
[✓] Enable ALU Opcode Pipeline Register	CLK0	CE0	RST0
[✓] Enable ALU Output and Flag Register	CLK0	CE0	RST0

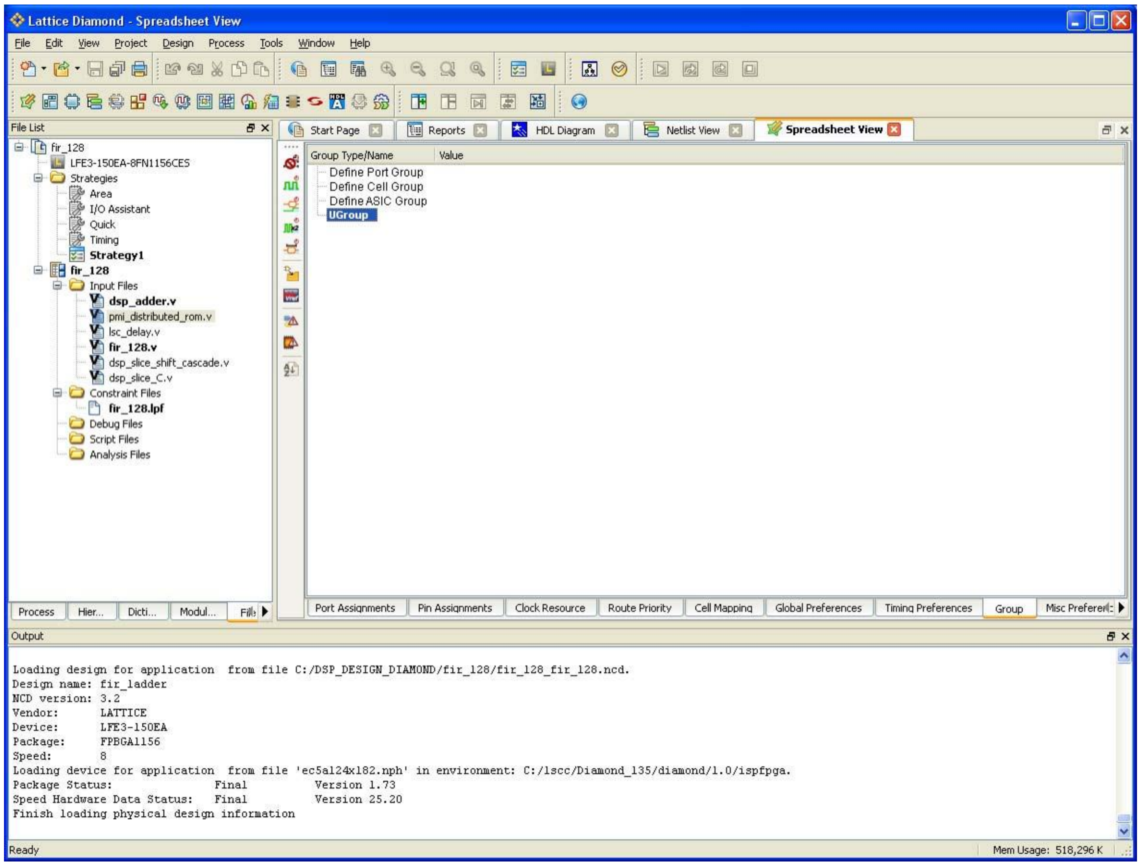
Figure B.13. SLICE Module – Register Setup

B.2. sysDSP in Diamond

Pre-Mapped View

Grouping Instances

sysDSP instances can be viewed or grouped together. In Diamond, select **Tools > Spreadsheet View**. Select the Groups tab. Right-click on **UGROUP** and select **New UGroup…**

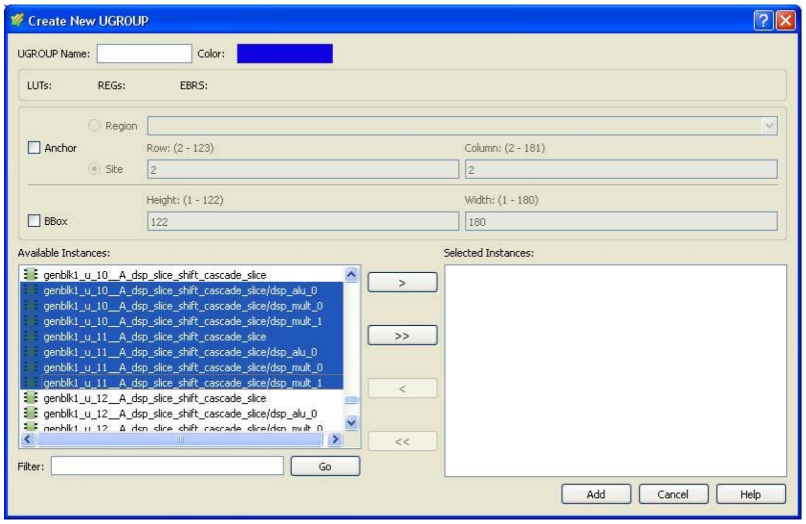


```
Loading design for application  from file C:/DSP_DESIGN_DIAMOND/fir_128/fir_128_fir_128.ncd.
Design name: fir_ladder
NCD version: 3.2
Vendor:      LATTICE
Device:      LFE3-150EA
Package:     FPBGA1156
Speed:       8
Loading device for application  from file 'ec5a124x182.nph' in environment: C:/lscc/Diamond_135/diamond/1.0/ispfpga.
Package Status:              Final          Version 1.73
Speed Hardware Data Status:  Final          Version 25.20
Finish loading physical design information
```

Figure B.14. Setting UGroups

A Create New UGroup window appears as shown in Figure B.15.

Left-click on an instance and select the right arrow (>) to add to the UGroup. To select multiple instances, hold the **Ctrl** or **Shift** key while selecting the instances with the left mouse button. Select the **Add** button to complete the UGroup.

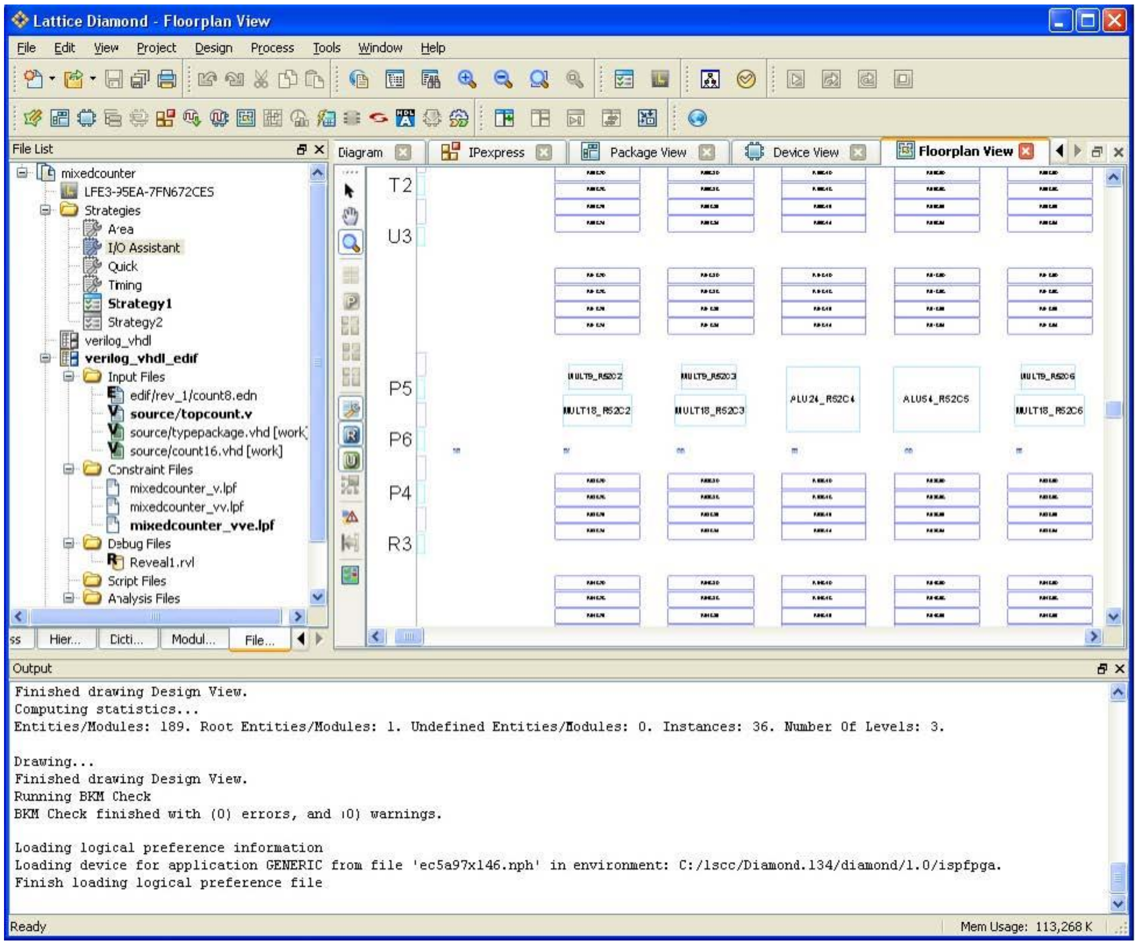


Create New UGROUP
UGROUP Name:
Color:
LUTs:
REGs:
EBRS:
Region
Anchor
Row: (2 - 123)
Column: (2 - 181)
Site
2
2
Height: (1 - 122)
Width: (1 - 180)
BBox
122
180
Available Instances:
Selected Instances:
genblk1_u_10__A_dsp_slice_shift_cascade_slice
genblk1_u_10__A_dsp_slice_shift_cascade_slice/dsp_alu_0
genblk1_u_10__A_dsp_slice_shift_cascade_slice/dsp_mult_0
genblk1_u_10__A_dsp_slice_shift_cascade_slice/dsp_mult_1
genblk1_u_11__A_dsp_slice_shift_cascade_slice
genblk1_u_11__A_dsp_slice_shift_cascade_slice/dsp_alu_0
genblk1_u_11__A_dsp_slice_shift_cascade_slice/dsp_mult_0
genblk1_u_11__A_dsp_slice_shift_cascade_slice/dsp_mult_1
genblk1_u_12__A_dsp_slice_shift_cascade_slice
genblk1_u_12__A_dsp_slice_shift_cascade_slice/dsp_alu_0
>
>>
<
<<
Filter:
Go
Add
Cancel
Help

Figure B.15. Selecting and Creating UGroups

Floorplan View in Diamond

The DSP slices are organized in rows, as shown in Figure B.16. It can be seen that each slice extends to four columns with the multipliers on the left and ALUs on the right.



```
Finished drawing Design View.
Computing statistics...
Entities/Modules: 189. Root Entities/Modules: 1. Undefined Entities/Modules: 0. Instances: 36. Number Of Levels: 3.

Drawing...
Finished drawing Design View.
Running BKM Check
BKM Check finished with (0) errors, and (0) warnings.

Loading logical preference information
Loading device for application GENERIC from file 'ec5a97x146.nph' in environment: C:/lscc/Diamond.134/diamond/1.0/ispfpga.
Finish loading logical preference file
```

Figure B.16. Floorplan View in Diamond

References

- LatticeECP3 Family Devices web page
- Boards, Demos, IP Cores, and Reference Designs for LatticeECP3 Family Devices web page
- Lattice Diamond Software web page
- Lattice Insights for Lattice Semiconductor Training Series and Learning Plans

Technical Support Assistance

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

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.4, March 2024

Section	Change Summary
All	• Changed document number from *TN1182* to *FPGA-TN-02193*. • Updated document template.
Disclaimers	Added this section.
Acronyms in This Document	Added this section.
References	Added this section.
Technical Support Assistance	Added the link to Lattice Answer Database.

Revision 1.3, February 2012

Section	Change Summary
All	Updated document with new corporate logo.

Revision 1.2, June 2010

Section	Change Summary
All	Updated for Lattice Diamond software support.

Revision 1.1, June 2009

Section	Change Summary
All	• Updated the performance table. • Updated the MAP, PAR and Trace reports. • Updated the DSP slice block diagram and description on arithmetic operation.
IPexpress Slice Module	Updated the IPexpress modules GUI and description.

Revision 1.0, February 2009

Section	Change Summary
All	Initial release.