# DSP Arithmetic Modules - Lattice Radiant Software

# User Guide

FPGA-IPUG-02050-1.5

December 2023

## Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

# Contents

# Figures

# Tables

# Acronyms in This Document

A list of acronyms used in this document.

| Acronym | Definition |
|---|---|
| FPGA | Field-Programmable Gate Array |
| IP | Intellectual Property |
| RTL | Register Transfer Level |

# 1. Introduction

The IP Catalog provides a variety of modules to assist your design work. These modules cover a variety of common functions and can be customized. They are optimized for Lattice FPGA devices built on the Lattice Nexus™ platform. Use these modules to speed up your design work and to produce the most effective results.

This guide describes the DSP Arithmetic modules that comes with the IP Catalog. These modules serve as the building block to realize a more complex or user-defined function. The descriptions mainly cover the ports, attributes and sample functional timing diagrams of the modules.

# 2. DSP Arithmetic Modules

## 2.1. Multiplier

A two-input multiplier performs signed/unsigned multiplication of the data from inputs data_a_i and data_b_i. The output result_o carries the Product of the multiplication operation.

The shaded portions of the Multiplier Schematic Diagram, as shown in Figure 2.1, are optional and are removed/ignored in certain configurations. Function formula:

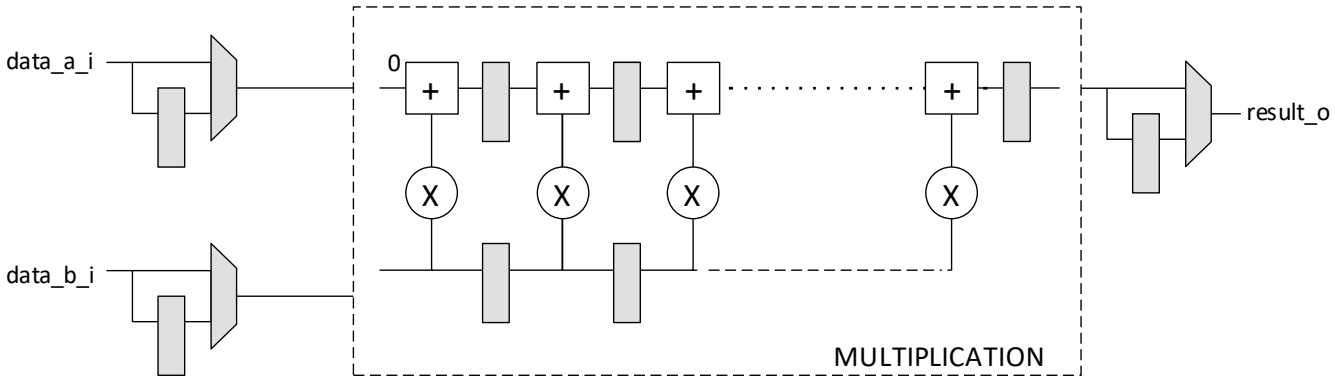


**Figure 2.1. Multiplier Schematic Diagram**

### 2.1.1. Multiplier Operation

**Table 2.1. Multiplier Operation**

| Configuration | Operation |
|---|---|
| Default | result_o = data_a_i*data_b_i |

### 2.1.2. Ports

**Table 2.2. Multiplier Ports**

| Signal Name | Direction | Width (bits) | Description |
|---|---|---|---|
| clk_i | IN | 1 | This signal connects to the clock pin of the input, and output<br>Unused when neither Input A, Input B or Output is set to Registered |
| ce_a_i | IN | 1 | This signal is an active HIGH synchronous clock enable for input A<br>1'b0 – retain the previous state<br>1'b1 – toggle on the rising edge of the clock as per the logic<br>Unused when Input A is not set to Registered. |
| ce_b_i | IN | 1 | This signal is an active HIGH synchronous clock enable for input B<br>1'b0 – retain the previous state<br>1'b1 – toggle on the rising edge of the clock as per the logic<br>Unused when Input B is not set to Registered |
| ce_o_i | IN | 1 | This signal is an active HIGH synchronous clock enable for Output<br>1'b0 – retain the previous state<br>1'b1 – toggle on the rising edge of the clock as per the logic<br>Unused when Output is not set to Registered |

| Signal Name | Direction | Width (bits) | Description |
|---|---|---|---|
| rst_a_i | IN | 1 | This signal initializes the flops of data_a_i in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Unused when Input A is not set to Registered. Reset mode depends on RST_MODE. |
| rst_b_i | IN | 1 | This signal initializes the flops of data_b_i in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Unused when Input B is not set to Registered. Reset mode depends on RST_MODE. |
| rst_o_i | IN | 1 | This signal initializes the flops of result_o in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Unused when Output is not set to Registered. Reset mode depends on RST_MODE. |
| data_a_i | IN | A_WIDTH | This signal contains an input for multiplication operation. |
| data_b_i | IN | B_WIDTH | This signal contains an input for multiplication operation. |
| result_o | OUT | OUT_WIDTH | This signal carries the product value of the multiplication.<br>Core functionality is represented below.<br>data_a_i × data_b_i |

### 2.1.3. Attributes

**Table 2.3. Multiplier Attributes**

| Configuration | Range | Default Value | Description |
|---|---|---|---|
| A_WIDTH | 2–72 | 9 | This configuration controls the width of the input data_a. |
| B_WIDTH | 2–72 | 9 | This configuration controls the width of the input data_b. |
| OUT_WIDTH | 4-144 | 18 | This parameter is uneditable and only gets the sum of A_WIDTH and B_WIDTH.<br>InputA Width + InputB Width = Result Width |
| RST_MODE | SYNC or ASYNC | ASYNC | This configuration controls the reset mode, either asynchronous reset or synchronous reset. When set to asynchronous reset, the reset value occurs at the preceding clock edge. |
| A_SIGNED | Signed, Unsigned | Signed | This configuration controls the sign interpretation of the input data_a_i, either signed or unsigned. |
| B_SIGNED | Signed, Unsigned | Signed | This configuration controls the sign interpretation of the input data_b_i, either signed or unsigned. |
| USE_IREGA | on, off | off | This configuration controls the registering of input data_a_i before routing them to the multiplication operation.<br>off – Unregistered. Inputs are directly routed.<br>on – Registered. Inputs are registered before routing. |
| USE_IREGB | on, off | off | This configuration controls the registering of the inputs, data_b_i before routing them to the multiplication operation.<br>off – Unregistered. Inputs are directly routed.<br>on – Registered. Inputs are registered before routing. |

| Configuration | Range | Default Value | Description |
|---|---|---|---|
| USE_OREG | on, off | off | This configuration controls the registering of the multiplier result onto the output, result_o.<br>off – Unregistered. Result is directly routed to the output.<br>on – Registered. Result is registered at the output. |
| **Internal Parameters** | | | |
| A_WDT | — | — | A_WDT = A_WIDTH |
| B_WDT | — | — | B_WDT = B_WIDTH |

## 2.1.4. Timing Diagram

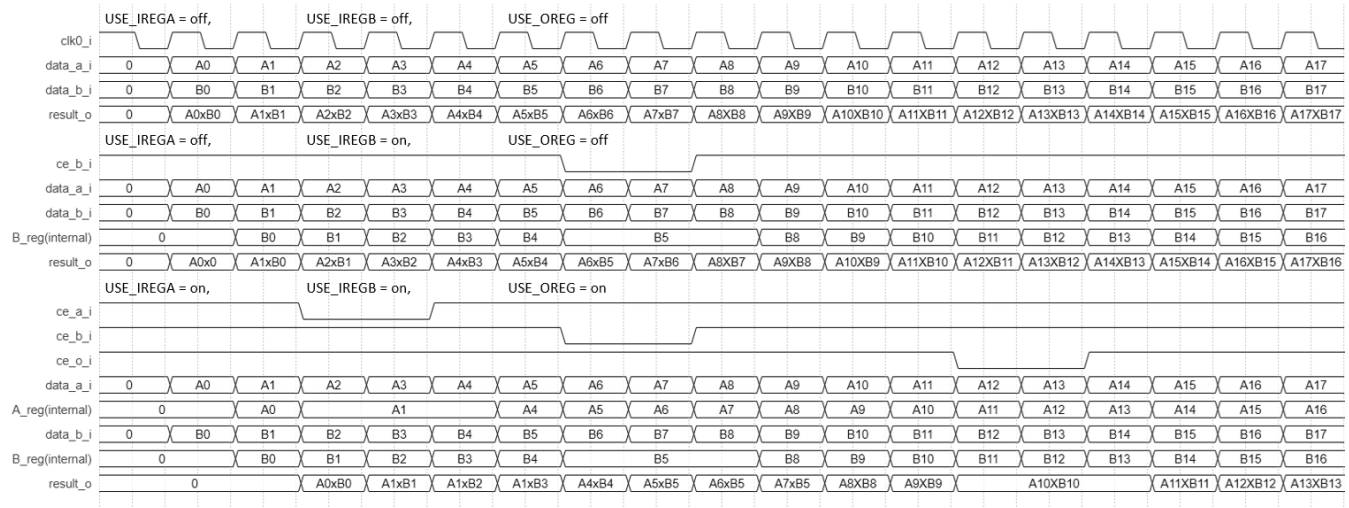


**Figure 2.2. Multiplier Timing Diagram**

**Table 2.4. Multiplier Timing Table**

| Attribute | | Latency from Input to Output |
|---|---|---|
| USE_IREG* | USE_OREG | |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 2 |

## 2.2. Multiply-Accumulate

A two-input multiplier with accumulate performs signed/unsigned multiplication of the data from inputs data_a_i and data_b_i and accumulates the result. The output result_o carries the Accumulation of Products.

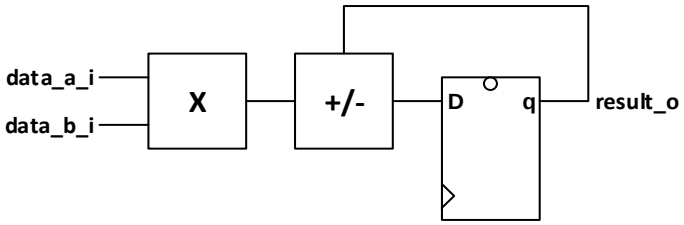


**Figure 2.3. Multiply-Accumulate Schematic**

### 2.2.1. Multiply-Accumulate Operation

**Table 2.5. Multiply-Accumulate Operations**

| Operation Dependency | Operation |
|---|---|
| addnsub_i = 0, accumsload = 0 | result_o = result_prev + (data_a_i × data_b_i) |
| addnsub_i = 0, accumsload = 1 | result_o = ld + (data_a_i × data_b_i) |
| addnsub_i = 1, accumsload = 0 | result_o = result_prev - (data_a_i × data_b_i) |
| addnsub_i = 1, accumsload = 1 | result_o = ld - (data_a_i × data_b_i) |

### 2.2.2. Ports

**Table 2.6. Multiply-Accumulate Ports**

| Signal Name | Direction | Width (bits) | Description |
|---|---|---|---|
| clk_i | IN | 1 | This signal connects to the clock pin of the input, and output<br>Unused when neither Input A, Input B or Output is set to Registered |
| ce_a_i | IN | 1 | This signal is an active HIGH synchronous clock enable for input A<br>1'b0 – retain the previous state<br>1'b1 – toggle on the rising edge of the clock as per the logic<br>Unused when Input A is not set to Registered. |
| ce_b_i | IN | 1 | This signal is an active HIGH synchronous clock enable for input B<br>1'b0 – retain the previous state<br>1'b1 – toggle on the rising edge of the clock as per the logic<br>Unused when Input B is not set to Registered |
| ce_p_i | IN | 1 | This signal is an active HIGH synchronous clock enable for pipeline register<br>1'b0 – retain the previous state<br>1'b1 – toggle on the rising edge of the clock as per the logic<br>Unused when Pipeline Register is not set to Registered |
| ce_o_i | IN | 1 | This signal is an active HIGH synchronous clock enable for Output<br>1'b0 – retain the previous state<br>1'b1 – toggle on the rising edge of the clock as per the logic<br>Unused when Output is not set to Registered |

| Signal Name | Direction | Width (bits) | Description |
|---|---|---|---|
| rst_a_i | IN | 1 | This signal initializes the flops of data_a_i in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Unused when Input A is not set to Registered. Reset mode depends on RST_MODE. |
| rst_b_i | IN | 1 | This signal initializes the flops of data_b_i in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Unused when Input B is not set to Registered. Reset mode depends on RST_MODE. |
| rst_p_i | IN | 1 | This signal initializes the flops of the pipelined signal in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Unused when Pipelined Register is not set to Registered. Reset mode depends on RST_MODE. |
| rst_o_i | IN | 1 | This signal initializes the flops of result_o in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Unused when Output is not set to Registered. Reset mode depends on RST_MODE. |
| data_a_i | IN | A_WIDTH | This signal contains the multiplicand value of the multiplication. |
| data_b_i | IN | B_WIDTH | This signal contains the multiplier value of the multiplication. |
| accumsload | IN | 1 | This signal destates if the ld input or the previews result_o is added/subtracted from the product of the two multipliers. |
| ld | IN | ACC_WIDTH | This signal is added or subtracted from the product of the two multiplier when accumsload is high. |
| addnsub_i | IN | 1 | This signal dictates the operation to be used by the accumulator – *1'b0* for addition and *1'b1* for subtraction. |
| result_o | OUT | ACC_WIDTH | This signal carries the accumulated value of the product of inputs data_a_i and data_b_i. Core functionality is represented below.<br>$Result = LD + (DataA_i \times DataB_i)$ ; accumsload = 1, addnsub_i = 0<br>$Result = LD - (DataA_i \times DataB_i)$ ; accumsload = 1, addnsub_i = 1<br>$Result = Result_{prev} + (DataA_i \times DataB_i)$ ; accumsload = 1, addnsub_i = 0<br>$Result = Result_{prev} - (DataA_i \times DataB_i)$ ; accumsload = 1, addnsub_i = 1 |

## 2.2.3. Attributes

**Table 2.7. Multiply-Accumulate Attributes**

| Configuration | Range | Default Value | Description |
|---|---|---|---|
| A_WIDTH | 2–72 | 9 | This configuration controls the width of the input data_a_i. |
| B_WIDTH | 2–72 | 9 | This configuration controls the width of the input data_b_i. |
| ACC_WIDTH | (A_WIDTH+ B_WIDTH+1) | 19 | This configuration controls the accumulator width. |

| Configuration | Range | Default Value | Description |
|---|---|---|---|
| A_SIGNED | Signed, Unsigned | Unsigned | This configuration controls the sign interpretation of the input data_a_i, either signed or unsigned. |
| B_SIGNED | Signed, Unsigned | Unsigned | This configuration controls the sign interpretation of the input data_b_i, either signed or unsigned. |
| RST_MODE | SYNC or ASYNC | ASYNC | This configuration controls the reset mode, either asynchronous reset or synchronous reset. When set to asynchronous reset, the reset value occurs at the preceding clock edge. |
| USE_IREGA | on, off | off | This configuration controls the registering of the inputs, data_a_i before routing them to the multiplication operation.<br>off – Unregistered. Inputs are directly routed.<br>on – Registered. Inputs are registered before routing. |
| USE_IREGB | on, off | off | This configuration controls the registering of the inputs, data_b_i before routing them to the multiplication operation.<br>off – Unregistered. Inputs are directly routed.<br>on – Registered. Inputs are registered before routing. |
| USE_PREG | on, off | off | This configuration controls the insertion of pipeline into the multiplier operation. |
| USE_OREG | on | on | This configuration controls the registering of the result onto the output, result_o. This is not editable. Result is registered at the output. |

## 2.2.4. Timing Diagram

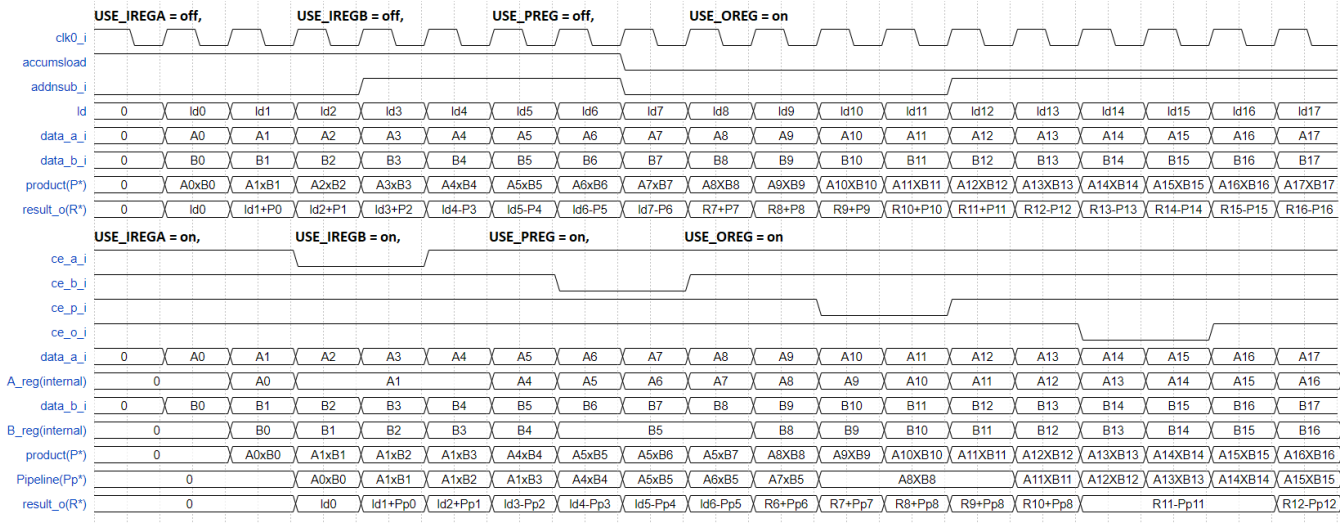


**Figure 2.4. Multiply-Accumulate Timing Waveforms**

**Table 2.8. Multiply-Accumulate Timing Table**

| Attribute | | | Latency from Input to Output |
|---|---|---|---|
| USE_IREG* | USE_PREG | USE_OREG | |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 2 |
| 1 | 0 | 1 | 2 |

| Attribute | | | Latency from Input to Output |
|---|---|---|---|
| USE_IREG* | USE_PREG | USE_OREG | |
| 1 | 1 | 1 | 3 |

## 2.3. Multiply-Add-Subtract

A pair of two-input multipliers that performs signed/unsigned multiplication of the data from inputs data_a and data_b with addition/subtraction on the product pair. The output result_o carries the Sum/Difference of Products.

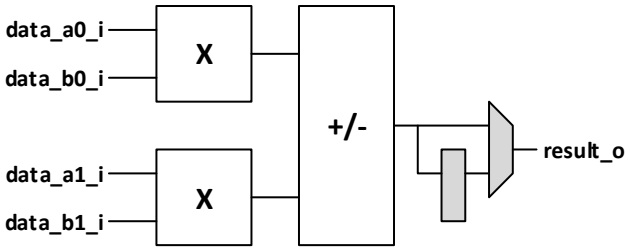


**Figure 2.5. Multiply-Add-Subtract Schematic**

### 2.3.1. Multiply-Add-Subtract Operation

**Table 2.9. Multiply-Add-Subtract Operations**

| Operation Dependency | Operation |
|---|---|
| addnsub_i = 0 | result_o = (data_a0_i × data_b0_i) + (data_a1_i × data_b1_i) |
| addnsub_i = 1 | result_o = (data_a0_i × data_b0_i) - (data_a1_i × data_b1_i) |

### 2.3.2. Ports

**Table 2.10. Multiply-Add-Subtract Ports**

| Signal Name | Direction | Width (bits) | Description |
|---|---|---|---|
| clk_i | IN | 1 | This signal connects to the clock pin of the input, and output<br>Unused when neither Input A, Input B or Output is set to Registered |
| ce_a_i | IN | 1 | This signal is an active HIGH synchronous clock enable for input A<br>1'b0 – retain the previous state<br>1'b1 – toggle on the rising edge of the clock as per the logic<br>Unused when Input A is not set to Registered. |
| ce_b_i | IN | 1 | This signal is an active HIGH synchronous clock enable for input B<br>1'b0 – retain the previous state<br>1'b1 – toggle on the rising edge of the clock as per the logic<br>Unused when Input B is not set to Registered |
| ce_p_i | IN | 1 | This signal is an active HIGH synchronous clock enable for pipeline register<br>1'b0 – toggles the previous state<br>1'b1 – toggle on the rising edge of the clock as per the logic<br>Unused when Pipeline Register is not set to Registered |
| ce_o_i | IN | 1 | This signal is an active HIGH synchronous clock enable for Output<br>1'b0 – retain the previous state<br>1'b1 – toggle on the rising edge of the clock as per the logic<br>Unused when Output is not set to Registered |

| Signal Name | Direction | Width (bits) | Description |
|---|---|---|---|
| rst_a_i | IN | 1 | This signal initializes the flops of data_a_i in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Unused when Input A is not set to Registered. Reset mode depends on RST_MODE. |
| rst_b_i | IN | 1 | This signal initializes the flops of data_b_i in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Unused when Input B is not set to Registered. Reset mode depends on RST_MODE. |
| rst_p_i | IN | 1 | This signal initializes the flops of the pipelined signal in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Unused when Pipelined Register is not set to Registered. Reset mode depends on RST_MODE. |
| rst_o_i | IN | 1 | This signal initializes the flops of result_o in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Unused when Output is not set to Registered. Reset mode depends on RST_MODE. |
| data_a0_i | IN | A_WIDTH | This signal contains the multiplicand value of the multiplication pair data_a0_i - data_b0_i. |
| data_a1_i | IN | A_WIDTH | This signal contains the multiplicand value of the multiplication pair data_a1_i - data_b1_i. |
| data_b0_i | IN | B_WIDTH | This signal contains the multiplier value of the multiplication pair data_a0_i - data_b0_i. |
| data_b1_i | IN | B_WIDTH | This signal contains the multiplier value of the multiplication pair data_b0_i - data_b1_i. |
| addnsub_i | IN | 1 | This signal dictates the operation to be used by the accumulator – *1'b0* for addition and *1'b1* for subtraction. |
| result_o | OUT | ACC_WIDTH | This signal carries the result_o of the addition/subtraction of the multiplication products.<br>Core functionality is represented below.<br>M0 = (data_a0_i × data_b0_i)<br>M1 = (data_a1_i × data_b1_i)<br>addnsub_i == 0: {M0 + M1}<br>addnsub_i == 1: {M0 – M1} |

### 2.3.3. Attributes

**Table 2.11. Multiply-Add-Subtract Attributes**

| Configuration | Range | Default Value | Description |
|---|---|---|---|
| A_WIDTH | 2–72 | 9 | This configuration controls the width of the inputs data_a0_i and data_a1_i. |
| B_WIDTH | 2–72 | 9 | This configuration controls the width of the inputs data_b0_i and data_b1_i. |

| Configuration | Range | Default Value | Description |
|---|---|---|---|
| ACC_WIDTH | 5-145 | 19 | This parameter is uneditable and only gets the sum of A_WIDTH, B_WIDTH + 1.<br>Input A Width + Input B Width + 1 = Result Width |
| A_SIGNED | Signed, Unsigned | Unsigned | This configuration controls the sign interpretation of the input data_a0_i and data_a1_i, either signed or unsigned. |
| B_SIGNED | Signed, Unsigned | Unsigned | This configuration controls the sign interpretation of the input data_b0_i and data_b1_i, either signed or unsigned. |
| RST_MODE | SYNC or ASYNC | SYNC | This configuration controls the reset mode, either asynchronous reset or synchronous reset. When set to asynchronous reset, the reset value occurs at the preceding clock edge. |
| USE_IREGAB0 | on, off | on | This configuration controls the registering of the inputs, data_a0_i and data_b0_i before routing them to the multiplication operation.<br>off – Unregistered. Inputs are directly routed.<br>on – Registered. Inputs are registered before routing. |
| USE_IREGAB1 | on, off | on | This configuration controls the registering of the inputs, data_a1_i and data_b1_i before routing them to the multiplication operation.<br>off – Unregistered. Inputs are directly routed.<br>on – Registered. Inputs are registered before routing. |
| USE_OREG | on, off | on | This configuration controls the registering of the output, result_o.<br>off – Unregistered. Result is directly routed to the output.<br>on – Registered. Result is registered at the output. |
| USE_PREG | on, off | on | This configuration controls the insertion of pipeline into the multiplier operation. |

### 2.3.4. Timing Diagram

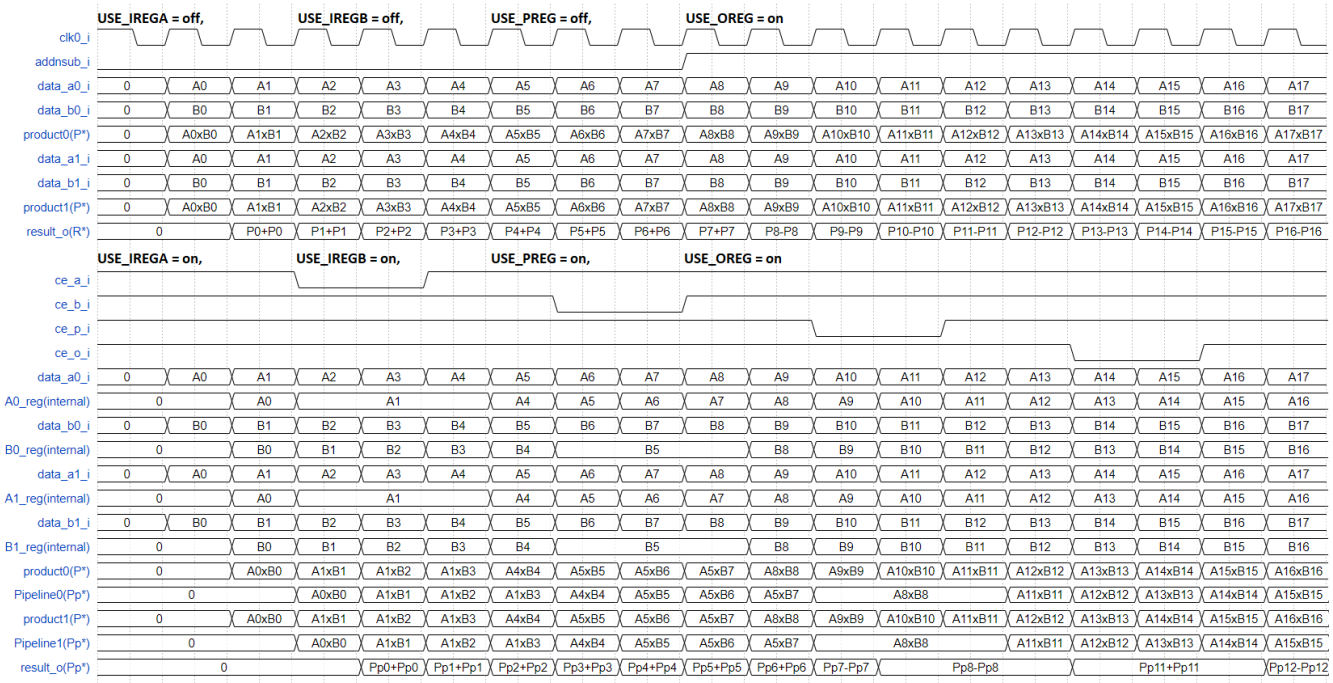


**Figure 2.6. Multiply-Add-Subtract Timing Diagram**

**Table 2.12. Multiply-Add-Subtract Timing Table**

| Attribute | | | Latency from Input to Output |
|---|---|---|---|
| USE_IREG* | USE_PREG | USE_OREG | |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 2 |
| 1 | 0 | 1 | 2 |
| 1 | 1 | 0 | 2 |
| 1 | 1 | 1 | 3 |

## 2.4. Multiply-Add-Subtract-Sum

Two pair of two-input multipliers that performs signed/unsigned multiplication of the data from inputs data_a and data_b with addition/subtraction on the product pair. The output result_o carries the Sumation of the results from the result of the added/subtracted procut pair.

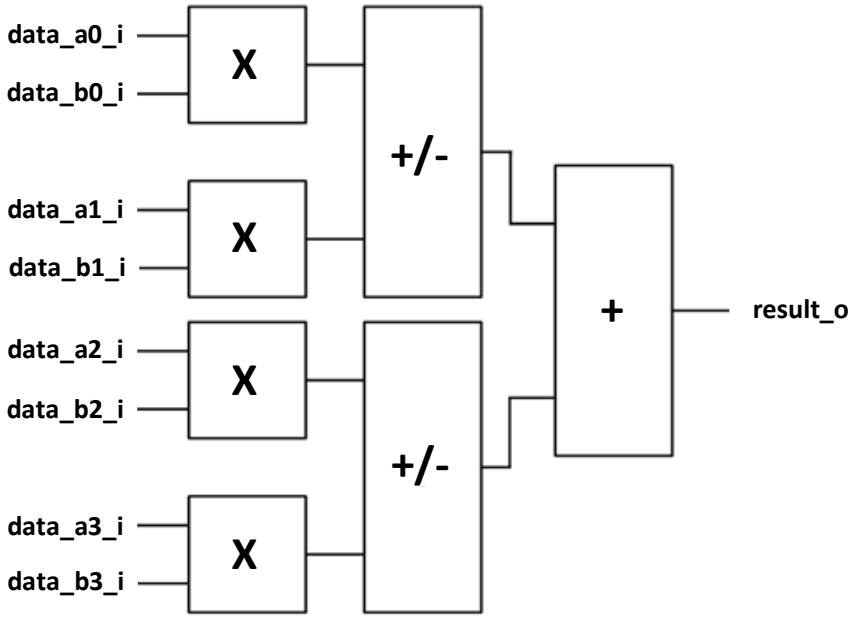


**Figure 2.7. Multiply-Add-Subtract-Sum Schematic**

### 2.4.1. Multiply-Add-Subtract-Sum Operation

**Table 2.13. Multiply-Add-Subtract-Sum Operation**

| Operation Dependency | Operation |
|---|---|
| addnsub0_i = 0<br>addnsub1_i = 0 | result_o = {(data_a0_i × data_b0_i) + (data_a1_i × data_b1_i) } + {(data_a2_i × data_b2_i) + (data_a3_i × data_b3_i)} |
| addnsub0_i = 1<br>addnsub1_i = 1 | result_o = {(data_a0_i × data_b0_i) - (data_a1_i × data_b1_i)} + {(data_a2_i × data_b2_i) - (data_a3_i × data_b3_i)} } |

### 2.4.2. Ports

**Table 2.14. Multiply-Add-Subtract-Sum Ports**

| Signal Name | Direction | Width (bits) | Description |
|---|---|---|---|
| clk_i | IN | 1 | This signal connects to the clock pin of the input, and output<br>Unused when neither Input A, Input B or Output is set to Registered |
| ce_a_i | IN | 1 | This signal is an active HIGH synchronous clock enable for input A<br>1'b0 – retain the previous state<br>1'b1 – toggle on the rising edge of the clock as per the logic<br>Un-used when Input A is not set to Registered. |
| ce_b_i | IN | 1 | This signal is an active HIGH synchronous clock enable for input B<br>1'b0 – retain the previous state<br>1'b1 – toggle on the rising edge of the clock as per the logic<br>Un-used when Input B is not set to Registered |
| ce_p_i | IN | 1 | This signal is an active HIGH synchronous clock enable for pipeline register<br>1'b0 – retain the previous state<br>1'b1 – toggle on the rising edge of the clock as per the logic<br>Un-used when Pipeline Register is not set to Registered |

| Signal Name | Direction | Width (bits) | Description |
|---|---|---|---|
| ce_o_i | IN | 1 | This signal is an active HIGH synchronous clock enable for Output<br>1'b0 – retain the previous state<br>1'b1 – toggle on the rising edge of the clock as per the logic<br>Un-used when Output is not set to Registered |
| rst_a_i | IN | 1 | This signal initializes the flops of data_a_i in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Un-used when Input A is not set to Registered. Reset mode depends on RST_MODE. |
| rst_b_i | IN | 1 | This signal initializes the flops of data_b_i in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Un-used when Input B is not set to Registered. Reset mode depends on RST_MODE. |
| rst_p_i | IN | 1 | This signal initializes the flops of the pipelined signal in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Un-used when Pipelined Register is not set to Registered. Reset mode depends on RST_MODE. |
| rst_o_i | IN | 1 | This signal initializes the flops of result_o in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Un-used when Output is not set to Registered. Reset mode depends on RST_MODE. |
| data_a0_i | IN | A_WIDTH | This signal contains the multiplicand value of the multiplication pair data_a0_i - data_b0_i. |
| data_a1_i | IN | A_WIDTH | This signal contains the multiplicand value of the multiplication pair data_a1_i - data_b1_i. |
| data_a2_i | IN | A_WIDTH | This signal contains the multiplicand value of the multiplication pair data_a2_i - data_b2_i. |
| data_a3_i | IN | A_WIDTH | This signal contains the multiplicand value of the multiplication pair data_a3_i - data_b3_i. |
| data_b0_i | IN | B_WIDTH | This signal contains the multiplier value of the multiplication pair data_a0_i - data_b0_i. |
| data_b1_i | IN | B_WIDTH | This signal contains the multiplier value of the multiplication pair data_a1_i - data_b1_i. |
| data_b2_i | IN | B_WIDTH | This signal contains the multiplier value of the multiplication pair data_a2_i - data_b2_i. |
| data_b3_i | IN | B_WIDTH | This signal contains the multiplier value of the multiplication pair data_a3_i - data_b3_i. |
| addnsub_i | IN | 1 | This signal dictates the operation to be used by the accumulator – *1'b0* for addition and *1'b1* for subtraction. |
| result_o | OUT | ACC_WIDTH | This signal carries the result_o of the addition/subtraction of the multiplication products.<br>Core functionality is represented below.<br>M0 = (data_a0_i × data_b0_i), M1 = (data_a1_i × data_b1_i),<br>M2 = (data_a2_i × data_b2_i), M3 = (data_a3_i × data_b3_i)<br>addnsub0_i == 0: {M0 + M1}, addnsub0_i == 1: {M0 – M1},<br>addnsub1_i == 0: {M2 + M3}, addnsub1_i == 1: {M2 – M3} |

### 2.4.3. Attributes

**Table 2.15. Multiply-Add-Subtract-Sum Attributes**

| Configuration | Range | Default Value | Description |
|---|---|---|---|
| A_WIDTH | 2–72 | 9 | This configuration controls the width of the inputs data_a0_i, data_a1_i, data_a2_i, and data_a3_i. |
| B_WIDTH | 2–72 | 9 | This configuration controls the width of the inputs data_b0_i, data_b1_i, data_b2_i, and data_b3_i. |
| ACC_WIDTH | 7-147 | 20 | This parameter is uneditable, it only gets the sum of A_WIDTH, B_WIDTH + 3.<br>Input A Width + Input B Width + 3 = Result Width |
| A_SIGNED | Signed, Unsigned | Unsigned | This configuration controls the sign interpretation of the input data_a0_i, data_a1_i, data_a2_i, and data_a3_i; either signed or unsigned. |
| B_SIGNED | Signed, Unsigned | Unsigned | This configuration controls the sign interpretation of the input data_b0_i, data_b1_i, data_b2_i, and data_b3_i; either signed or unsigned. |
| USE_IREGA | on, off | on | This configuration controls the registering of the inputs, data_a0_i, data_a1_i, data_a2_i, and data_a3_i before routing them to the multiplication operation.<br>off – Unregistered. Inputs are directly routed.<br>on – Registered. Inputs are registered before routing. |
| USE_IREGB | on, off | on | This configuration controls the registering of the inputs, data_b0_i, data_b1_i, data_b2_i, and data_b3_i before routing them to the multiplication operation.<br>off – Unregistered. Inputs are directly routed.<br>on – Registered. Inputs are registered before routing. |
| USE_OREG | on, off | on | This configuration controls the registering of the output, result_o.<br>off – Unregistered. Result is directly routed to the output.<br>on – Registered. Result is registered at the output. |
| USE_PREG | on, off | on | This configuration controls the insertion of pipeline into the multiplier operation. |
| RST_MODE | SYNC or ASYNC | SYNC | This configuration controls the reset mode, either asynchronous reset or synchronous reset. When set to asynchronous reset, the reset value occurs at the preceding clock edge. |

## 2.4.4. Timing Diagram

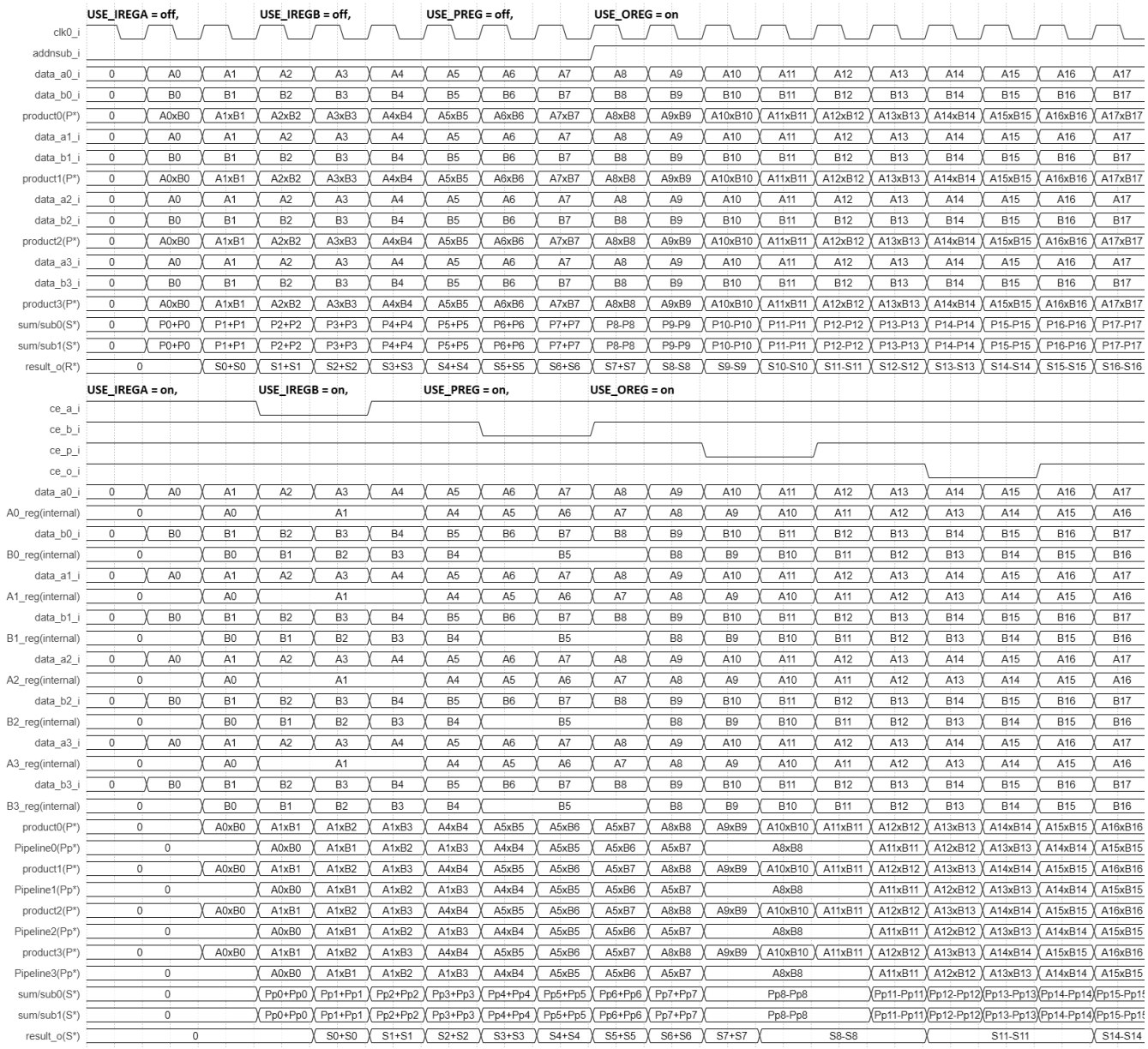


**Figure 2.8. Multiply-Add-Subtract-Sum Timing Diagram**

**Table 2.16. Multiply-Add-Subtract-Sum Timing Table**

| Attribute | | | Latency from Input to Output |
|---|---|---|---|
| USE_IREG* | USE_PREG | USE_OREG | |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 2 |
| 1 | 0 | 1 | 2 |
| 1 | 1 | 0 | 2 |

| Attribute | | | Latency from Input to Output |
|---|---|---|---|
| USE_IREG* | USE_PREG | USE_OREG | |
| 1 | 1 | 1 | 3 |

## 2.5. Multiply-Multiply-Accumulate

A pair of two-input multipliers that performs signed/unsigned multiplication of the data from inputs data_a* and data_b* with addition/subtraction of the product pair and an accumulator at the end. The output result_o carries the Sum/Difference of Products added to the previews result_o or ld input depending on the accumsload value.

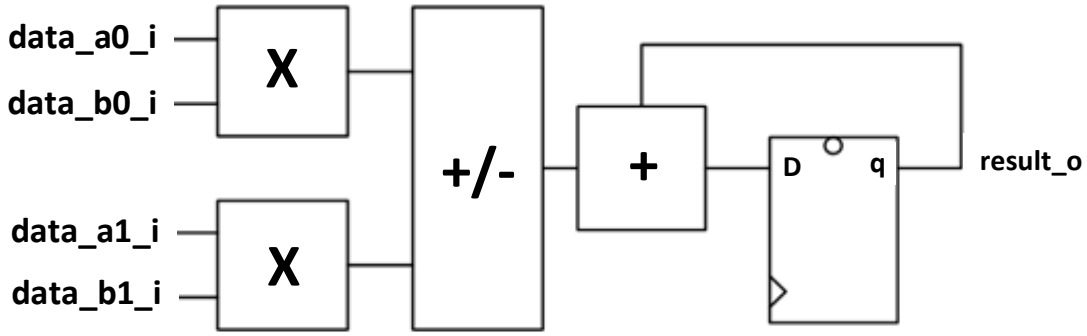


**Figure 2.9. Multiply-Multiply-Accumulate Schematic**

### 2.5.1. Multiply-Multiply-Accumulate Operation

**Table 2.17. Multiply-Multiply-Accumulate Operations**

| Operation Dependency | Operation |
|---|---|
| addnsub_i = 0, accumsload = 0 | result_o = result_nxt + (data_a0_i × data_b0_i) + (data_a1_i × data_b1_i) |
| addnsub_i = 0, accumsload = 1 | result_o = ld + (data_a0_i × data_b0_i) + (data_a1_i × data_b1_i) |
| addnsub_i = 1, accumsload = 0 | result_o = result_nxt + (data_a0_i × data_b0_i) - (data_a1_i × data_b1_i) |
| addnsub_i = 1, accumsload = 1 | result_o = ld + (data_a0_i × data_b0_i) - (data_a1_i × data_b1_i) |

### 2.5.2. Ports

**Table 2.18. Multiply-Multiply-Accumulate Ports**

| Signal Name | Direction | Width (bits) | Description |
|---|---|---|---|
| clk_i | IN | 1 | This signal connects to the clock pin of the input, and output<br>Un-used when neither Input A, Input B or Output is set to Registered |
| ce_a_i | IN | 1 | This signal is an active HIGH synchronous clock enable for input A<br>1'b0 - retain the previous state<br>1'b1 - toggle on the rising edge of the clock as per the logic<br>Un-used when Input A is not set to Registered. |

| Signal Name | Direction | Width (bits) | Description |
|---|---|---|---|
| ce_b_i | IN | 1 | This signal is an active HIGH synchronous clock enable for input B<br>1'b0 - retain the previous state<br>1'b1 - toggle on the rising edge of the clock as per the logic<br>Un-used when Input B is not set to Registered |
| ce_p_i | IN | 1 | This signal is an active HIGH synchronous clock enable for pipeline register<br>1'b0 - retain the previous state of the multipliers<br>1'b1 - toggle on the rising edge of the clock as per the logic<br>Un-used when Pipeline Register is not set to Registered |
| ce_o_i | IN | 1 | This signal is an active HIGH synchronous clock enable for Output<br>1'b0 - retain the previous state<br>1'b1 - toggle on the rising edge of the clock as per the logic<br>Un-used when Output is not set to Registered |
| rst_a_i | IN | 1 | This signal initializes the flops of data_a_i in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Un-used when Input A is not set to Registered. Reset mode depends on RST_MODE. |
| rst_b_i | IN | 1 | This signal initializes the flops of data_b_i in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Un-used when Input B is not set to Registered. Reset mode depends on RST_MODE. |
| rst_p_i | IN | 1 | This signal initializes the flops of the pipelined signal in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Un-used when Pipelined Register is not set to Registered. Reset mode depends on RST_MODE. |
| rst_o_i | IN | 1 | This signal initializes the flops of result_o in the design to a known state.<br>It is an active HIGH reset whose deassertion should be synchronized with the clock during integration. Un-used when Output is not set to Registered. Reset mode depends on RST_MODE. |
| data_a0_i | IN | A_WIDTH | This signal contains the multiplicand value of the multiplication pair data_a0_i - data_b0_i. |
| data_a1_i | IN | A_WIDTH | This signal contains the multiplicand value of the multiplication pair data_a1_i - data_b1_i. |
| data_b0_i | IN | B_WIDTH | This signal contains the multiplier value of the multiplication pair data_a0_i - data_b0_i. |
| data_b1_i | IN | B_WIDTH | This signal contains the multiplier value of the multiplication pair data_a1_i - data_b1_i. |
| accumsload | IN | 1 | This signal destates if the ld input or the previews result_o is added from the product of the two multipliers. |

| Signal Name | Direction | Width (bits) | Description |
|---|---|---|---|
| ld | IN | ACC_WIDTH | This signal is added or subtracted from the product of the two multiplier when accumsload is high. |
| addnsub_i | IN | 1 | This signal dictates the operation to be used on the product pair. '0' for addition and '1' for subtraction. |
| result_o | OUT | ACC_WIDTH | This signal carries the result_o of the addition/subtraction of the multiplication products added to the previews result_o.<br>Core functionality is represented below.<br>M0 = (data_a0_i × data_b0_i)<br>M1 = (data_a1_i × data_b1_i)<br>result_nxt = previews result_o<br>addnsub_i == 0: {M0 + M1}<br>addnsub_i == 1: {M0 – M1} |

### 2.5.3. Attributes

**Table 2.19. Multiply-Multiply-Accumulate Attributes**

| Configuration | Range | Default Value | Description |
|---|---|---|---|
| A_WIDTH | 2–72 | 9 | This configuration controls the width of the inputs data_a0_i and data_a1_i. |
| B_WIDTH | 2–72 | 9 | This configuration controls the width of the inputs data_b0_i and data_b1_i. |
| ACC_WIDTH | 7-147 | 21 | This parameter is uneditable, it only gets the sum A_WIDTH, B_WIDTH + 3.<br>Input A Width + Input B Width + 3 = Result Width |
| A_SIGNED | Signed, Unsigned | Unsigned | This configuration controls the sign interpretation of the input data_a0_i and data_a1_i, either signed or unsigned. |
| B_SIGNED | Signed, Unsigned | Unsigned | This configuration controls the sign interpretation of the input data_b0_i and data_b1_i, either signed or unsigned. |
| RST_MODE | SYNC or ASYNC | SYNC | This configuration controls the reset mode, either asynchronous reset or synchronous reset. |
| USE_IREGAB0 | On, Off | On | This configuration controls the registering of the inputs, data_a0_i and data_b0_i before routing them to the multiplication operation.<br>off – Unregistered. Inputs are directly routed.<br>on – Registered. Inputs are registered before routing. |
| USE_IREGAB1 | On, Off | On | This configuration controls the registering of the inputs, data_a1_i and data_b1_i before routing them to the multiplication operation.<br>off – Unregistered. Inputs are directly routed.<br>on – Registered. Inputs are registered before routing. |
| USE_PREG | On, Off | Off | This configuration controls the insertion of pipeline into the multiplier operation. |
| USE_OREG | On | On | Not Configurable |

## 2.5.4. Timing Diagram

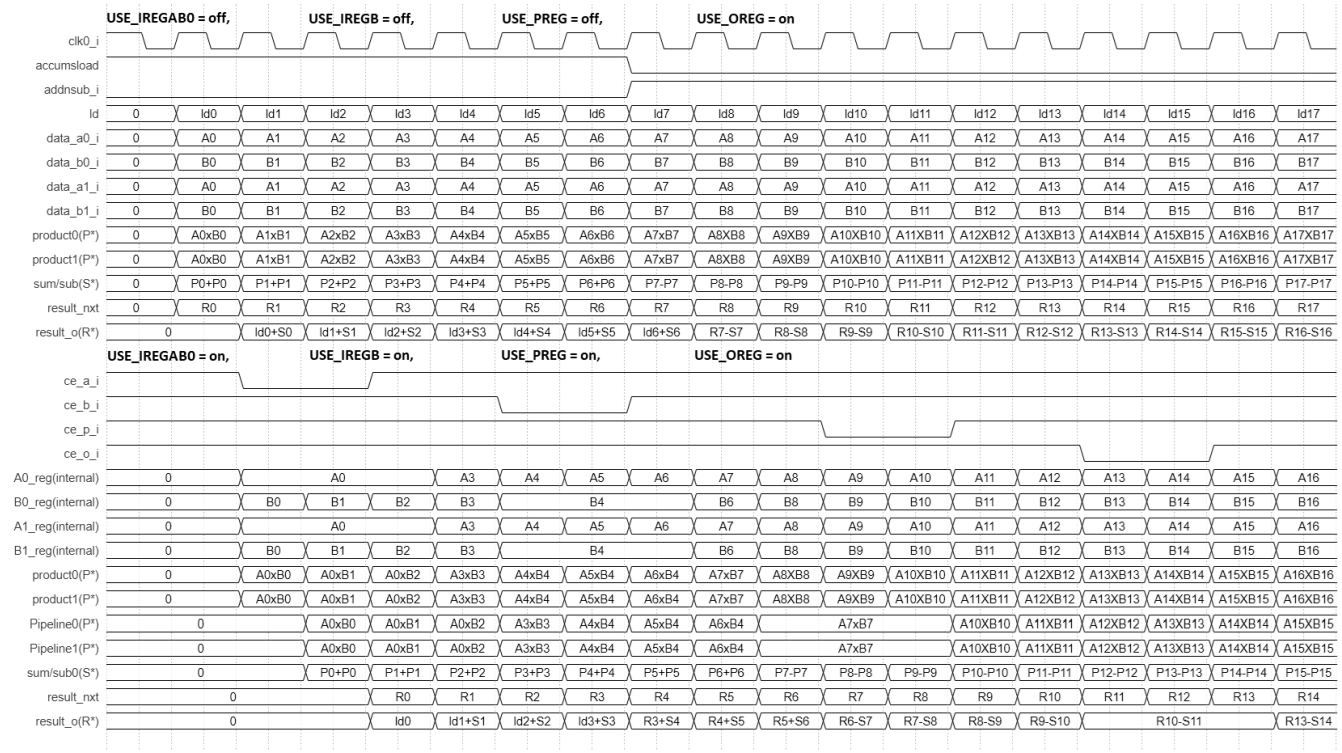


**Figure 2.10. Multiply-Multiply-Accumulate Timing Diagram**

**Table 2.20. Multiply-Multiply-Accumulate Timing Table**

| Attribute | | | Latency from Input to Output |
|---|---|---|---|
| USE_IREG* | USE_PREG | USE_OREG | |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 2 |
| 1 | 0 | 1 | 2 |
| 1 | 1 | 0 | 2 |
| 1 | 1 | 1 | 3 |

# 3. IP Generation

The Module/IP Block Wizard in Lattice Radiant™ Software allows you to generate, create, or open modules for the target device. From the Lattice Radiant Software, select the IP Catalog tab as shown in Figure 3.1.

The left pane of the IP Catalog window displays the module/IP tree. You can utilize this to specify a variety of arithmetic modules in your designs.

The available arithmetic modules in the Lattice Radiant Software IP catalog are:

- Multiplier
- Mult-Accumulate
- Mult-Add-Sub

The right pane of the window shows the description of the selected module and provides link(s) to the documentation.

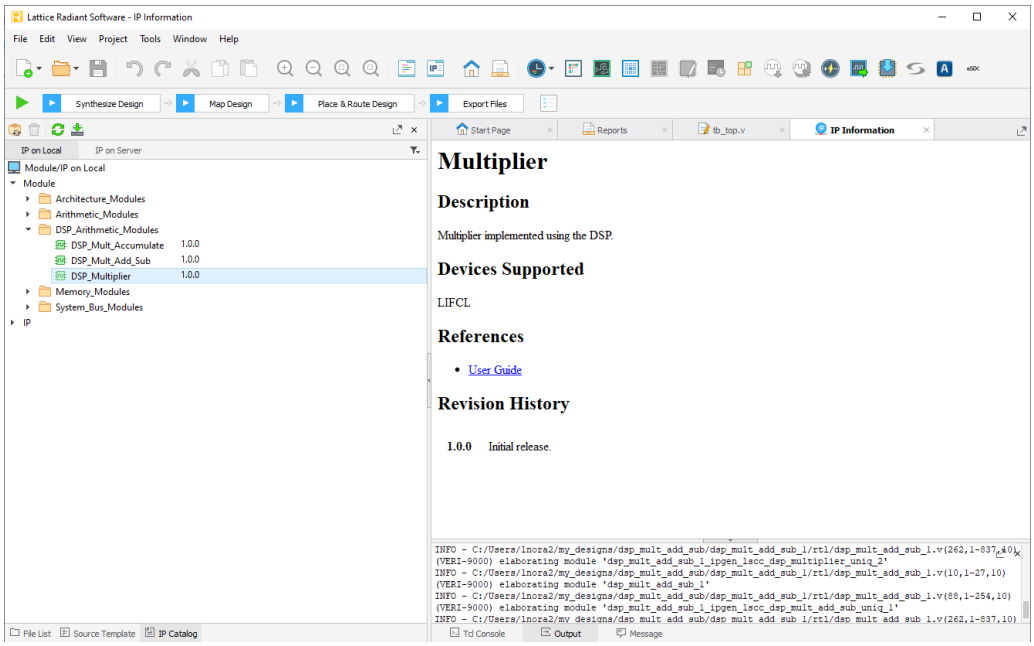


**Figure 3.1. DSP Arithmetic Modules**

## 3.1. Generating a 32-bit Multiplier Module

The following section shows an example of generating a 32-bit Multiplier module. This process is similar for all DSP Arithmetic Modules.

To generate a 32-bit Multiplier module:

1. Double-click Multiplier under DSP_Arithmetic_Modules. This opens the Module/IP Block Wizard.
2. Fill out the following information then click **Next**. An example is shown in Figure 3.3.
   - Language
   - Instance name
   - Create in

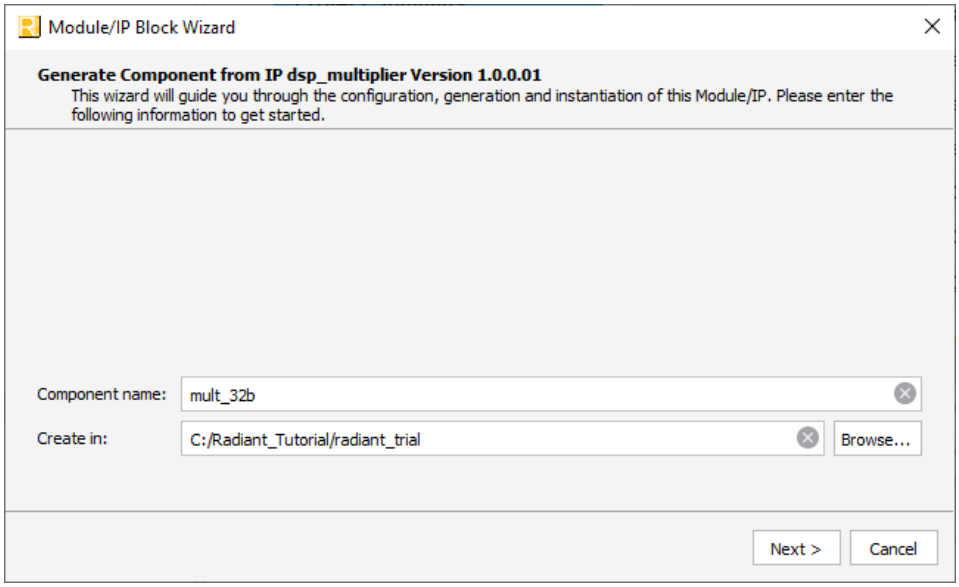


**Figure 3.2. Example: Generating 32-bit Multiplier Using Module/IP Block Wizard**

3. In the **IP Configuration** page, customize the Multiplier by selecting options as shown in Figure 3.3.

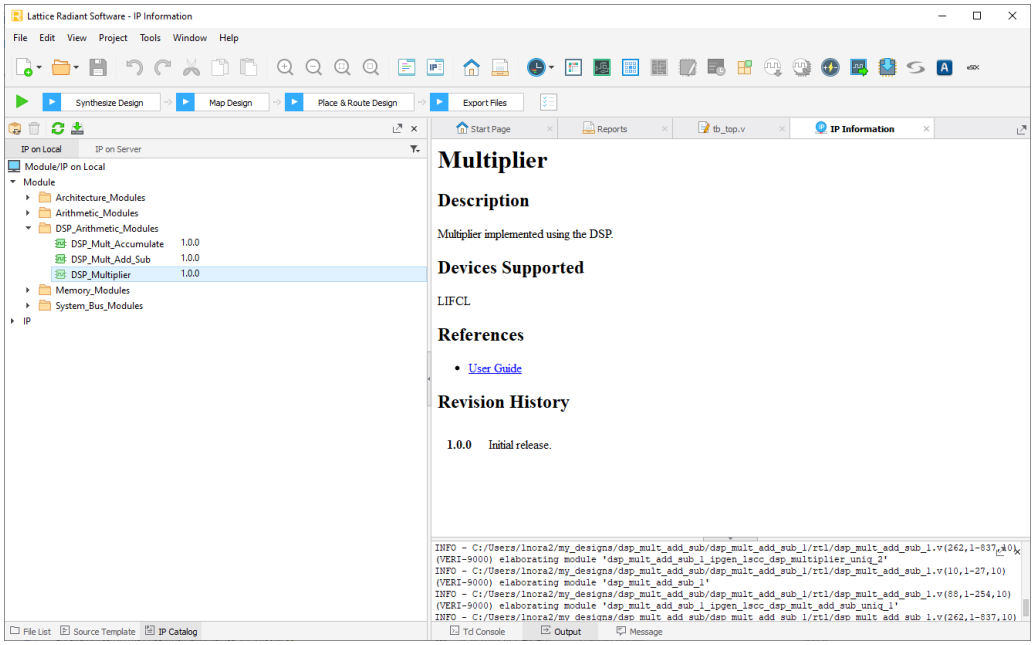


**Figure 3.3. Example: Generating 32-bit Multiplier in IP Configuration**

4. When all the options are set, click **Generate**.
5. Click **Finish**.

Once this module is in the Lattice Radiant Software project, it can be instantiated in other modules within the project. You can view the files and instance/s added in the **File List** tab.

# 4. Simulation Flow

The Lattice Radiant software also allows you to simulate configured arithmetic modules, so you can observe the conditions of the output with a given stimuli. This section details the individual steps needed to run simulation through Simulation Wizard. In the sample procedure, a Multiplier on default settings is generated, including instance name for the multiplier.

Table 4.1 provides a summary description of the files used in the project.

**Table 4.1. File List**

| File | Sim | Synthesis | Description |
|---|---|---|---|
| rlt/<Instance_Name>.v | Yes | Yes | Top Level RTL file with the selected configuration. The main IP file |
| testbench/dut_params | Yes | — | Top level parameters of the generated RTL file |
| testbench/dut_inst | Yes | — | Instantiated version of the <IP_name>.v file for simulation use |
| tb_top.v | Yes | — | Test bench template, this can be edited by the user to match their specific needs. |
| <Instance_Name>.cfg | — | — | This file contains the configuration options used to recreate or modify the core in the IP Platform. |
| <Instance_Name>.ipx | — | — | The IPX file holds references to all of the elements of an IP or Module after it is generated from the IP Platform GUI. The file is used to bring in the appropriate files during the design implementation and analysis. It is also used to re-load parameter settings into the IP Platform when the IP/Module is being regenerated. |
| component.xml | — | — | Contains the ipxact:component information of the IP. |
| design.xml | — | — | Documents the configuration attributes of the IP in IP-XACT 2014 format. |
| rtl/<Instance_Name>_bb.v | — | — | This file provides the synthesis black box. |
| misc/<Instance Name>_tmpl.v<br>misc /<Instance Name>_tmpl.vhd | — | — | These files provide instance templates for the module. |
| eval/constraint.pdc | — | — | This file contains the PDC constraints for this IP. Refer to section 4.24.2 for details on how to use this file. |

## 4.1. Generating a Multiplier on Default Settings

To generate a Multiplier on default settings:

1. First, create a new IP instance. Refer to IP Generation section for details on how to generate an IP in a project.

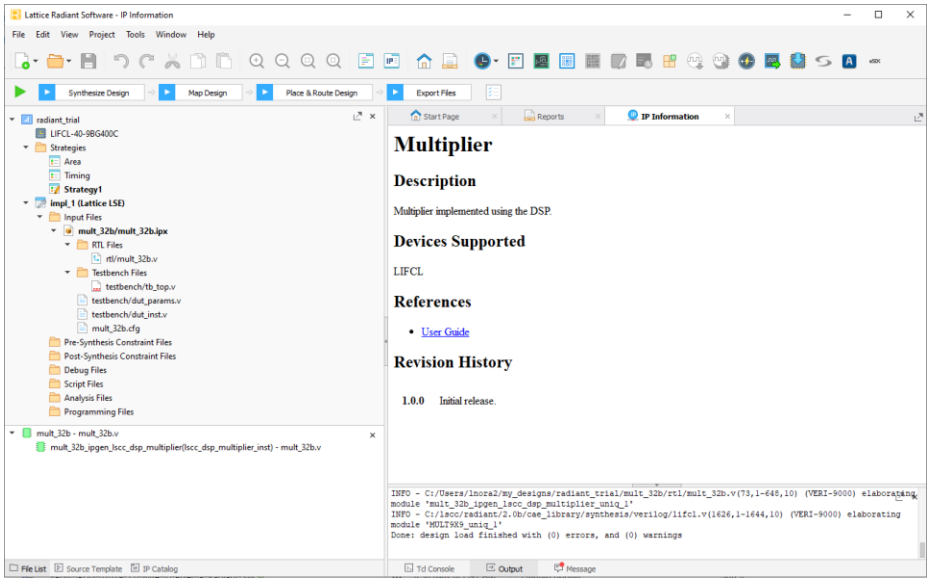


**Figure 4.1. Project with an Instance of Multiplier**

2. Right-click on the **impl_1** under **Strategies** folder and select **Add Existing File**. The file selection window opens showing the folder where the project is saved.

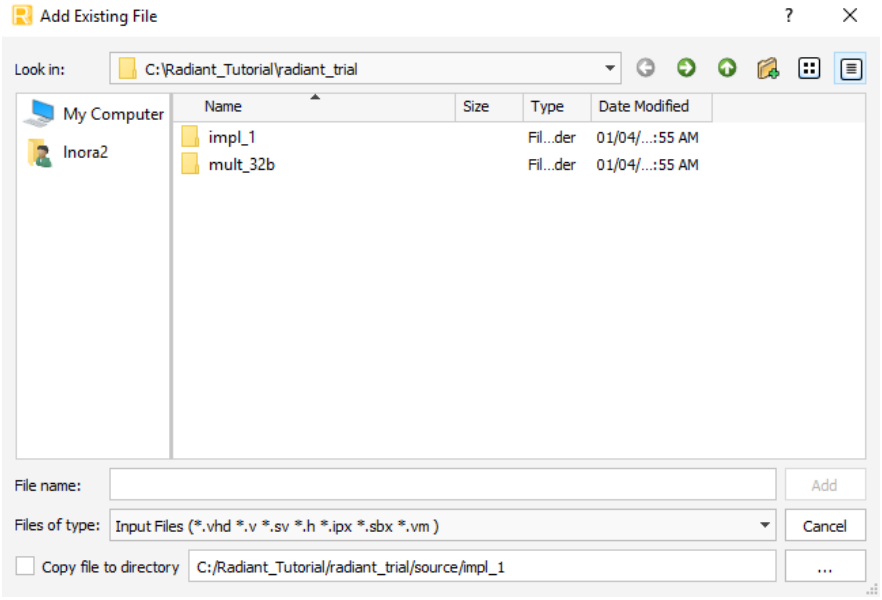


**Figure 4.2. File Selection Window**

3. Click the name of the IP instance, in this case, **mult_32b**.
4. Navigate to **<Instance_Name>/testbench** and click **tb_top.v.**
5. Click **Add**. This adds the top-level testbench to the project as shown in Figure 4.3 after opening.

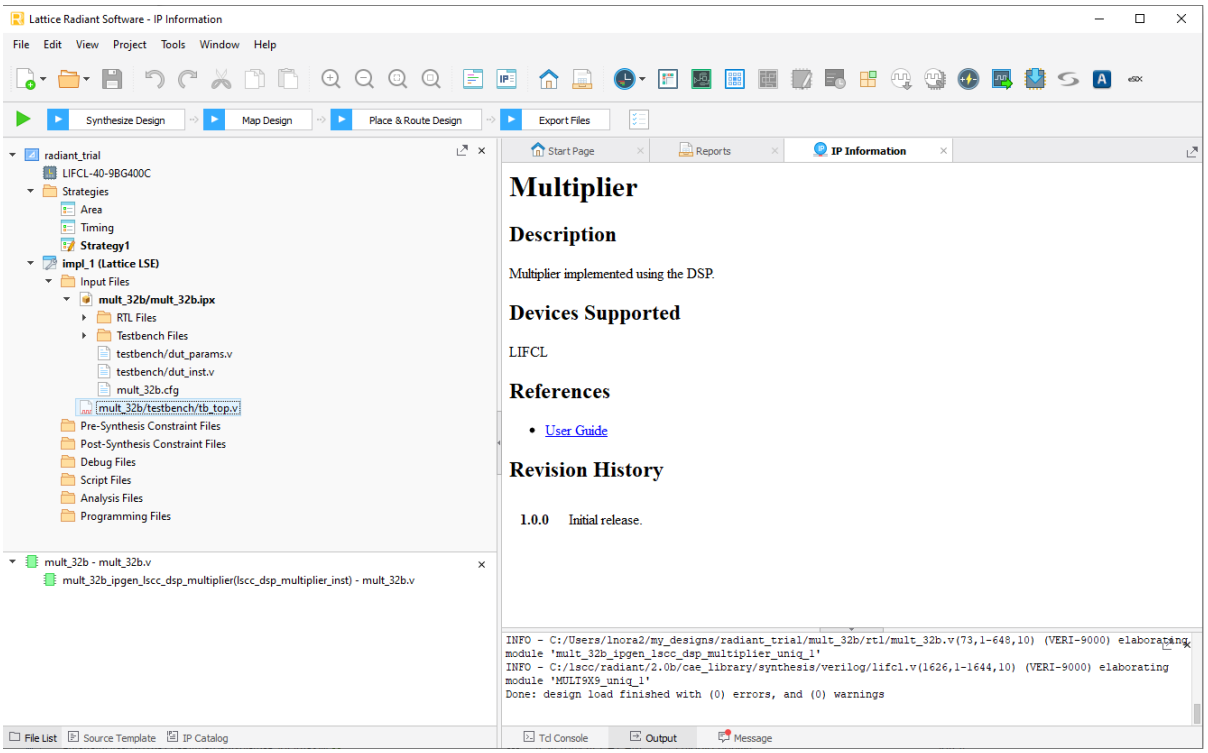


**Figure 4.3. Top Level Testbench Instance Added to the Project**

6. At this point, you can modify the contents of the testbench if you want add a specific stimuli or check a specific option. For this example, however, it is left on the default RTL file. To simulate the project, click on **Tools > Simulation Wizard**.
7. The **Simulation Wizard** window appears. Click **Next**. You are prompted to select the working folder and the test bench name. Enter **test**.

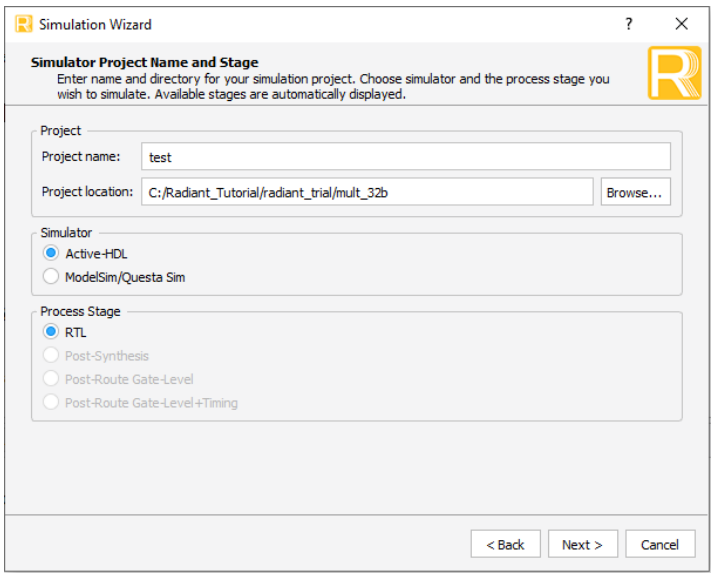


**Figure 4.4. Simulation Wizard**

8. Click **Next**.
9. In the prompt for folder creation, click **Yes**.
10. Moving to the processing stage, select **RTL**. Click **Next**.

11. The **Source Files** list appears. This shows the list of files included for simulation. Here, you can add or subtract any other files that are missed. Leave the file order for now. Click **Next**.

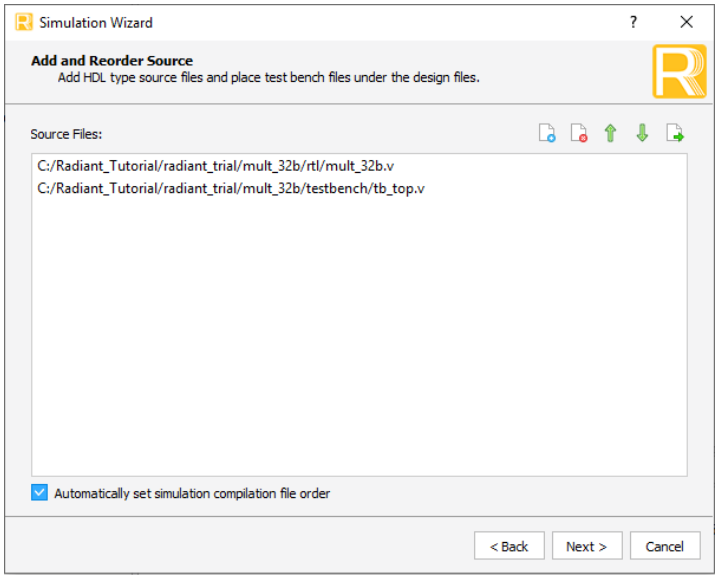


**Figure 4.5. Final Simulation Files**

12. In the next section, the files as well as the other dependencies are parsed to check for final errors. We can also select the top simulation module, before passing the files to a simulator for execution. Click **Next** for a summary view.

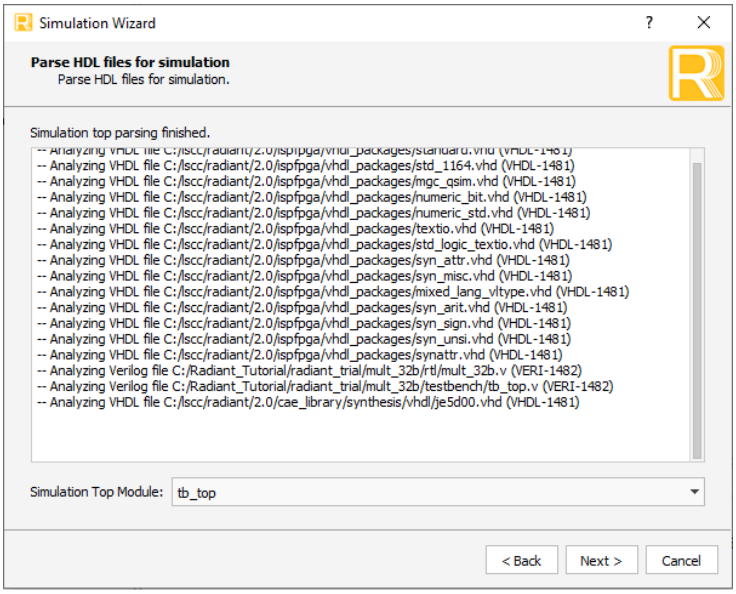


**Figure 4.6. File Parsing and Top Module Selection**

13. Click **Finish**. The ModelSim Lattice Edition software opens, which automatically compiles and runs the RTL simulation.

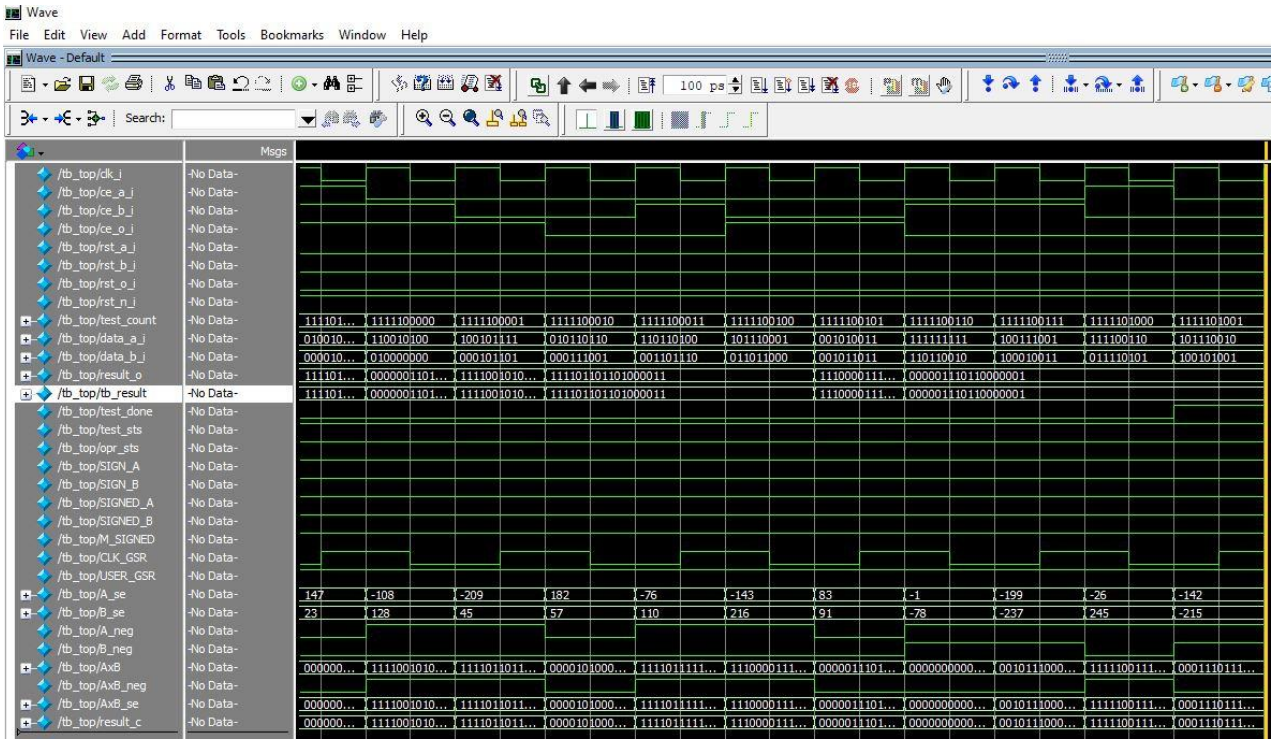


**Figure 4.7. Simulation Waveforms**

## 4.2. Constraining the IP

You need to provide proper timing and physical design constraints to ensure that your design meets the desired performance goals on the FPGA. Add the content of the following IP constraint file to your design constraints: <Instance_Path>/<Instance_Name>/eval/constraint.pdc

The above constraint file has been verified during IP evaluation with the IP instantiated directly at the top-level module. You can modify the constraints in this file with thorough understanding of the effect of each constraint.

To use this constraint file:

- Copy the contents of constraint.pdc to the top-level design constrain for post-synthesis.

Refer to Lattice Radiant Timing Constraints Methodology for details on how to constrain your design.

# Appendix A. Resource Utilization

This appendix provides resource utilization information for Lattice FPGAs using the DSP Arithmetic Modules. The IP configurations shown in this chapter were generated using the Lattice Radiant software.

## A.1. Multiplier

**Table A.1. Performance and Resource Utilization (LIFCL)**

| Configuration | Fmax (MHz) | LUTs | Registers | DSP | EBRs |
|---|---|---|---|---|---|
| 2-bit Signed Multiplier with its input and output registered | 200.000 | 0 | 0 | 0.5 | 0 |
| 9-bit Signed Multiplier with its input and output registered | 200.000 | 0 | 0 | 0.5 | 0 |
| 18-bit Signed Multiplier with its input and output registered | 200.000 | 1 | 0 | 1 | 0 |

**Note:** Performance and utilization characteristics are generated targeting an LIFCL-40-9BG400I device using Lattice Radiant Software 2023.1 and LSE Synthesis Tool. Performance may vary when using this IP in a different density, speed, or grade within the Lattice LIFCL-40 family or in a different software version. Fmax is generated when the FPGA design contains only the DSP Multiplier Module and the target frequency is 250 MHz.

**Table A.2. Performance and Resource Utilization (LFCPNX)**

| Configuration | Fmax (MHz) | LUTs | Registers | DSP | EBRs |
|---|---|---|---|---|---|
| 2-bit Signed Multiplier with its input and output registered | 200.000 | 0 | 0 | 0.5 | 0 |
| 9-bit Signed Multiplier with its input and output registered | 200.000 | 0 | 0 | 0.5 | 0 |
| 18-bit Signed Multiplier with its input and output registered | 200.000 | 1 | 0 | 1 | 0 |

**Note:** Performance and utilization characteristics are generated targeting an LFCPNX-100-9LFG672I device using Lattice Radiant Software 2023.1 and LSE Synthesis Tool. Performance may vary when using this IP in a different density, speed, or grade within the Lattice LFCPNX-100 family or in a different software version. Fmax is generated when the FPGA design contains only the DSP Multiplier Module and the target frequency is 250 MHz.

## A.2. Multiply-Accumulate

**Table A.3. Performance and Resource Utilization (LIFCL)**

| Configuration | Fmax (MHz) | LUTs | Registers | DSP | EBRs |
|---|---|---|---|---|---|
| 2-bit Signed Mult-Acc with input, output, and pipeline register | 200.000 | 1 | 0 | 1 | 0 |
| 9-bit Signed Mult-Acc (Addition) with input and output register using DSP blocks | 200.000 | 1 | 0 | 1 | 0 |
| 18-bit Signed Mult-Acc with input, output and pipeline register | 200.000 | 1 | 0 | 1 | 0 |

**Note:** Performance and utilization characteristics are generated targeting an LIFCL-40-9BG400I device using Lattice Radiant Software 2023.1 and LSE Synthesis Tool. Performance may vary when using this IP in a different density, speed, or grade within the Lattice LIFCL-40 family or in a different software version. Fmax is generated when the FPGA design contains only the DSP Multiplier Module and the target frequency is 250 MHz.

**Table A.4. Performance and Resource Utilization (LFCPNX)**

| Configuration | Fmax (MHz) | LUTs | Registers | DSP | EBRs |
|---|---|---|---|---|---|
| 2-bit Signed Mult-Acc with input, output, and pipeline register | 200.000 | 1 | 0 | 1 | 0 |
| 9-bit Signed Mult-Acc (Addition) with input and output register using DSP blocks | 200.000 | 1 | 0 | 1 | 0 |
| 18-bit Signed Mult-Acc with input, output and pipeline register | 200.000 | 1 | 0 | 4 | 0 |

**Note:** Performance and utilization characteristics are generated targeting an LFCPNX-100-9LFG672I device using Lattice Radiant Software 2023.1 and LSE Synthesis Tool. Performance may vary when using this IP in a different density, speed, or grade within the Lattice LFCPNX-100 family or in a different software version. Fmax is generated when the FPGA design contains only the DSP Multiplier Module and the target frequency is 250 MHz.

## A.3. Mult-Add-Subtract

**Table A.5. Performance and Resource Utilization (LIFCL)**

| Configuration | Fmax (MHz) | LUTs | Registers | DSP | EBRs |
|---|---|---|---|---|---|
| 2-bit Signed Mult-Add-Sub with input, output, and pipeline register | 200.000 | 1 | 0 | 2 | 0 |
| 9-bit Signed Mult-Add-Sub with input, output, and pipeline register | 200.000 | 1 | 0 | 2 | 0 |
| 18-bit Signed Mult-Add-Sub with output register | 200.000 | 76 | 39 | 2 | 0 |

**Note:** Performance and utilization characteristics are generated targeting an LIFCL-40-9BG400I device using Lattice Radiant Software 2023.1 and LSE Synthesis Tool. Performance may vary when using this IP in a different density, speed, or grade within the Lattice LIFCL-40 family or in a different software version. Fmax is generated when the FPGA design contains only the DSP Multiplier Module and the target frequency is 250 MHz.

**Table A.6. Performance and Resource Utilization (LFCPNX)**

| Configuration | Fmax (MHz) | LUTs | Registers | DSP | EBRs |
|---|---|---|---|---|---|
| 2-bit Signed Mult-Add-Sub with input, output, and pipeline register | 200.000 | 1 | 0 | 2 | 0 |
| 9-bit Signed Mult-Add-Sub with input, output, and pipeline register | 200.000 | 1 | 0 | 2 | 0 |
| 18-bit Signed Mult-Add-Sub with output register | 196.928 | 76 | 39 | 2 | 0 |

**Note:** Performance and utilization characteristics are generated targeting an LFCPNX-100-9LFG672I device using Lattice Radiant Software 2023.1 and LSE Synthesis Tool. Performance may vary when using this IP in a different density, speed, or grade within the Lattice LFCPNX-100 family or in a different software version. Fmax is generated when the FPGA design contains only the DSP Multiplier Module and the target frequency is 250 MHz.

## A.4. Mult-Add-Subtract-Sum

**Table A.7. Performance and Resource Utilization (LIFCL)**

| Configuration | Fmax (MHz) | LUTs | Registers | DSP | EBRs |
|---|---|---|---|---|---|
| 2-bit Signed Mult-Add-Sub-Sum with input, output, and pipeline register | 181.884 | 40 | 9 | 2 | 0 |
| 9-bit Signed Mult-Add-Sub-Sum with input, output, and pipeline register | 135.245 | 124 | 23 | 2 | 0 |

**Note:** Performance and utilization characteristics are generated targeting an LIFCL-40-9BG400I device using Lattice Radiant Software 2023.1 and LSE Synthesis Tool. Performance may vary when using this IP in a different density, speed, or grade within the Lattice LIFCL-40 family or in a different software version. Fmax is generated when the FPGA design contains only the DSP Multiplier Module and the target frequency is 250 MHz.

**Table A.8. Performance and Resource Utilization (LFCPNX)**

| Configuration | Fmax (MHz) | LUTs | Registers | DSP | EBRs |
|---|---|---|---|---|---|
| 2-bit Signed Mult-Add-Sub-Sum with input, output, and pipeline register | 189.916 | 40 | 9 | 2 | 0 |
| 9-bit Signed Mult-Add-Sub-Sum with input, output, and pipeline register | 151.630 | 124 | 23 | 2 | 0 |

**Note:** Performance and utilization characteristics are generated targeting an LFCPNX-100-9LFG672I device using Lattice Radiant Software 2023.1 and LSE Synthesis Tool. Performance may vary when using this IP in a different density, speed, or grade within the Lattice LFCPNX-100 family or in a different software version. Fmax is generated when the FPGA design contains only the DSP Multiplier Module and the target frequency is 250 MHz.

## A.5. Mult-Mult-Accumulate

**Table A.9. Performance and Resource Utilization (LIFCL)**

| Configuration | Fmax (MHz) | LUTs | Registers | DSP | EBRs |
|---|---|---|---|---|---|
| 9-bit Signed Mult-Mult-Accumulate with input, output, and pipeline register | 200.000 | 1 | 0 | 2 | 0 |
| 18-bit Signed Mult-Mult-Accumulate with input, output, and pipeline register | 188.359 | 159 | 80 | 2 | 0 |

**Note:** Performance and utilization characteristics are generated targeting an LIFCL-40-9BG400I device using Lattice Radiant Software 2023.1 and LSE Synthesis Tool. Performance may vary when using this IP in a different density, speed, or grade within the Lattice LIFCL-40 family or in a different software version. Fmax is generated when the FPGA design contains only the DSP Multiplier Module and the target frequency is 250 MHz.

**Table A.10. Performance and Resource Utilization (LFCPNX)**

| Configuration | Fmax (MHz) | LUTs | Registers | DSP | EBRs |
|---|---|---|---|---|---|
| 9-bit Signed Mult-Mult-Accumulate with input, output, and pipeline register | 200.000 | 1 | 0 | 2 | 0 |
| 18-bit Signed Mult-Mult-Accumulate with input, output, and pipeline register | 158.881 | 159 | 80 | 2 | 0 |

Note: Performance and utilization characteristics are generated targeting an LFCPNX-100-9LFG672I device using Lattice Radiant Software 2023.1 and LSE Synthesis Tool. Performance may vary when using this IP in a different density, speed, or grade within the Lattice LFCPNX-100 family or in a different software version. Fmax is generated when the FPGA design contains only the DSP Multiplier Module and the target frequency is 250 MHz.

# References

- Lattice Nexus Platform web page.
- Lattice Radiant Software web page.
- Lattice Insights for Lattice Semiconductor training courses and learning plans
- Lattice Radiant Timing Constraints Methodology

# 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

**Document Revision 1.5, December 2023**

| Section | Change Summary |
|---|---|
| Disclaimers | Updated this section. |
| Simulation Flow | • Replaced instances of “IP name” with “instance name”.<br>• Updated Table 4.1.<br>• Added section 4.2. Constraining the IP. |
| Resource Utilization | In Table A.1., Table A.2., Table A.3., Table A.4., Table A.5., Table A.6., Table A.7., Table A.8., Table A.9., Table A.10.: added an Fmax (MHz) column, updated LUTs, Registers, and DSP values, and updated the note below each table. |
| References | Added this section. |
| Technical Support Assistance | Added a link to the Lattice Answer Database. |

**Document Revision 1.4, Lattice Radiant SW Version 3.1, November 2021**

| Section | Change Summary |
|---|---|
| DSP Arithmetic Modules | Updated port definition for reset signals in Table 2.2, Table 2.6, Table 2.10, Table 2.14, and Table 2.18. |
| Appendix A. Resource Utilization | Updated section content to add tables for LFCPNX. |

**Document Revision 1.3, Lattice Radiant SW Version 2.0, June 2021**

| Section | Change Summary |
|---|---|
| Introduction | Indicated support for *Lattice FPGA devices built on the Lattice Nexus platform*. |
| Simulation Flow | • Revised information regarding the steps covered in the section.<br>• Updated the Generating a Multiplier on Default Settings section.<br>  • Updated steps 12 and 13.<br>  • Changed Figured 4.7.<br>  • Removed last paragraph. |

**Document Revision 1.2, Lattice Radiant SW Version 2.0, June 2020**

| Section | Change Summary |
|---|---|
| Introduction | Added Certus-NX as supported device. |
| DSP Arithmetic Modules | • Updated the following tables:<br>  • Table 2.3<br>  • Table 2.7<br>  • Table 2.11<br>  • Table 2.15<br>  • Table 2.19 |

**Document Revision 1.1, Lattice Radiant SW Version 2.0, February 2020**

| Section | Change Summary |
|---|---|
| DSP Arithmetic Modules | • Updated the following tables and figures:<br>  • Figure 2.4<br>  • Figure 2.6<br>  • Figure 2.8<br>  • Table 2.5<br>  • Table 2.12<br>  • Table 2.13<br>• Added new section Multiply-Multiply-Accumulate. |

| Section | Change Summary |
|---|---|
| Appendix A. Resource Utilization | Added new section Mult-Mult-Accumulate. |

**Document Revision 1.0, Lattice Radiant SW Version 2.0, November 2019**

| Section | Change Summary |
|---|---|
| All | Changed document status from Preliminary to final. |
| DSP Arithmetic Modules | Removed ADD_SUB parameter from the tables below and set the signal for operation addnsub_i as dynamic.<br>• Table 2.5. Multiply-Accumulate Ports<br>• Table 2.8. Multiply-Add-Subtract Ports<br>• Table 2.11. Multiply-Add-Subtract-Sum Ports |
| Appendix A. Resource Utilization | Removed Fmax (MHz) column from the tables. |

**Document Revision 0.80, Lattice Radiant SW Version 2.0, September 2019**

| Section | Change Summary |
|---|---|
| All | Preliminary release. |


www.latticesemi.com