Avant Adder Tree Module - Lattice Radiant Software

User Guide

FPGA-IPUG-02185-1.2

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 Adder Tree Module is used to add large numbers of digits at one time. The module is pipelined and can operate on each clock cycle.

1.1. Features

The key features of Adder Tree Module include:

- Configurable data width
- Supports multiple number of inputs
- Configurable Reset Mode
- Supports Enable/Disable of Fully Pipeline Mode
- Supports Enable/Disable of Input and Output Registers

1.2. Conventions

1.2.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL.

1.2.2. Signal Names

Signal names that end with:

- _*n* are active low
- _*i* are input signals
- _*o* are output signals

1.2.3. Attribute Names

Attribute names in this document are formatted in title case and italicized (*Attribute Name*).

2. Functional Description

The Adder Tree Module is a pipelined adder which implements "*data0_i + data1_i + ... + datan_i = result_o*" function.

Figure 2.1 shows the Adder Tree block diagram.

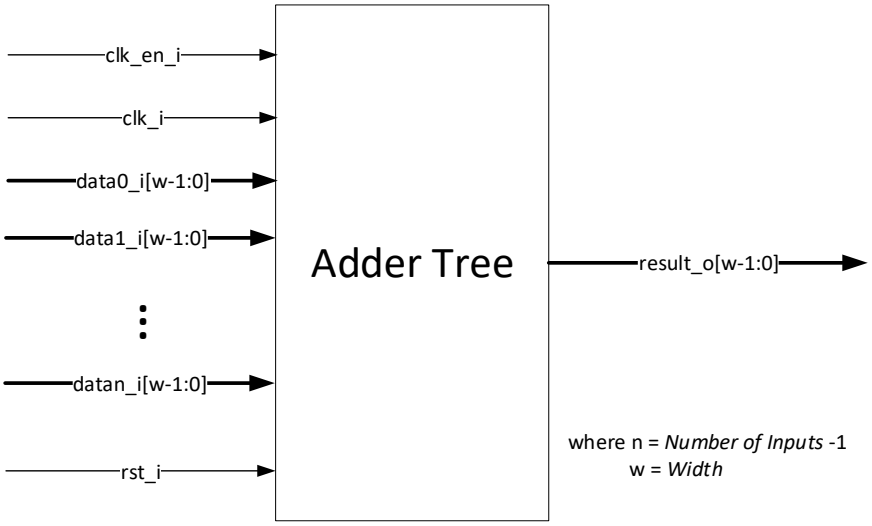

Figure 2.1. Adder Tree Block Diagram

Figure 2.2 shows the implementation diagram of Adder Tree when input register is enabled. The input registers used are internal to the DSP instantiated on the design. The output latency is 1.

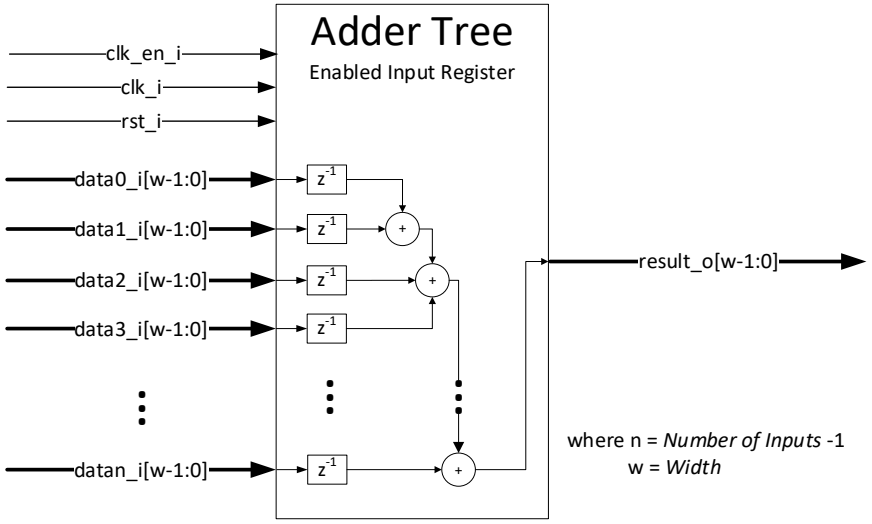

Figure 2.2. Adder Tree Enabled Input Register Implementation Diagram

Figure 2.3 shows the implementation diagram of Adder Tree when output register is enabled. The output register used is internal to the DSP instantiated on the design. The output latency is 1.

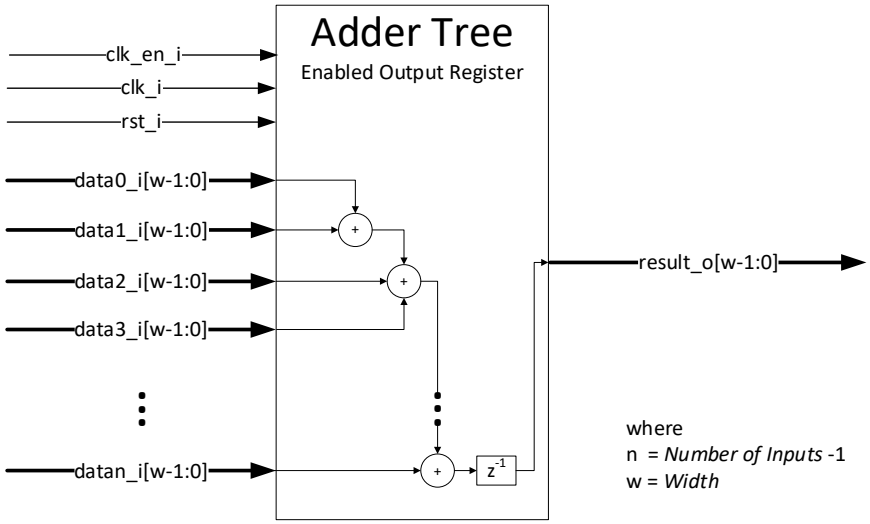

Figure 2.3. Adder Tree Enabled Output Register Implementation Diagram

Figure 2.4 shows the implementation diagram of Adder Tree when input and output register are enabled. The registers used are internal to the DSP instantiated on the design. The output latency is 2.

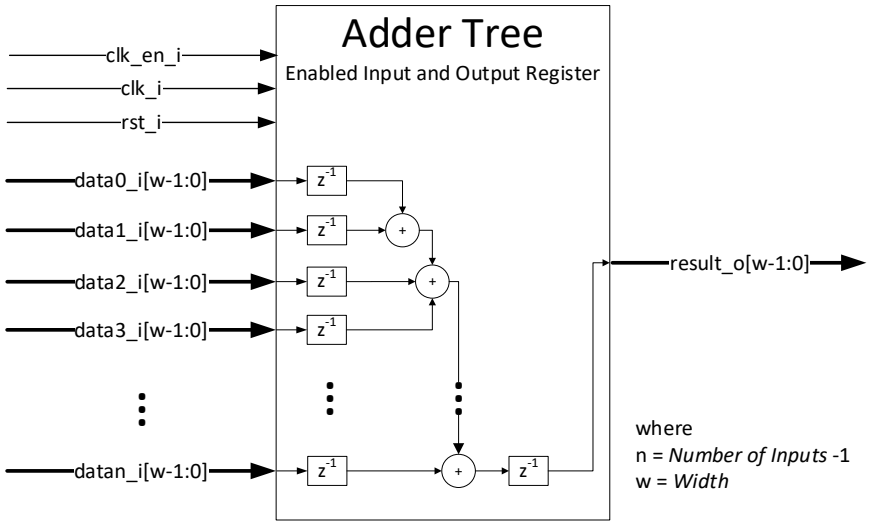

Figure 2.4. Adder Tree Enabled Input and Output Register Implementation Diagram

Figure 2.5 shows the implementation diagram of Adder Tree when output register and fully pipelined mode are enabled. During this configuration, fabric-based flip-flops are used on the implementation to align inputs to registered DSP outputs. The number of FFs used can be calculated by:

- Number of Slices for Even Number of Inputs = Number of Inputs / 2
- Number of Slices for Odd Number of Inputs = (Number of Inputs + 1) / 2
- Number of Flip-Flops = Width * (Number of Slices – 1) * (Number of Slices)

The output latency is Number of Slices + 1.

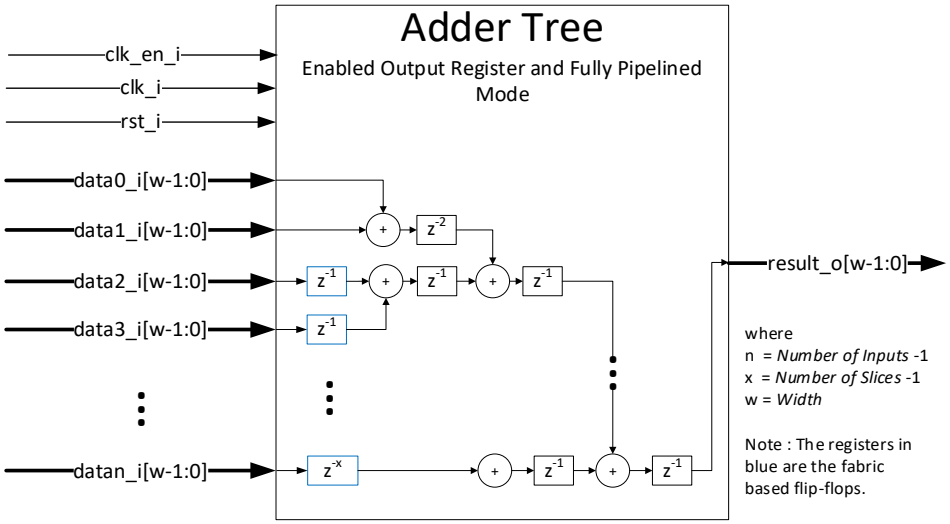

Figure 2.5. Adder Tree Enabled Output Register and Fully Pipelined Mode Implementation Diagram

Figure 2.6 shows the implementation diagram of Adder Tree when input and output register and fully pipelined mode are enabled. During this configuration, internal DSP register and fabric-based flip-flops are used on the implementation to align inputs to registered DSP outputs. The number of FFs used can be calculated same as when output register and fully pipelined mode 1D filter is used. The output latency is Number of Slices + 2.

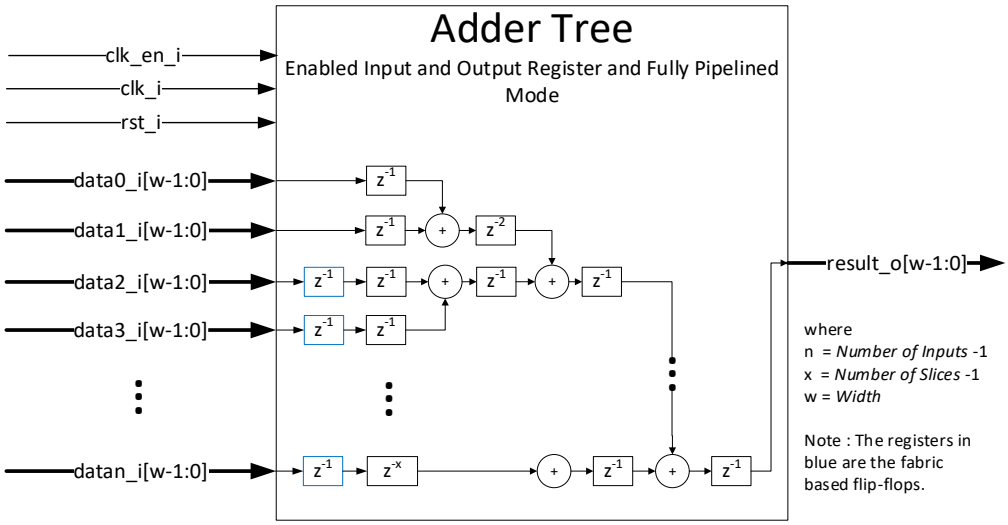

Figure 2.6. Adder Tree Enabled Input and Output Register and Fully Pipelined Mode Implementation Diagram

2.1. Signal Descriptions

Table 2.1. Adder Tree Module Signal Description

Port Name	I/O	Width	Description
Clock and Reset Ports			
clk_i	In	1	System clock input. Available if any of the following attributes is/are enabled: *Enable Fully Pipelined Mode, Enable Input Register or Enable Output Register.*
rst_i	In	1	Reset input. Available if any of the following attributes is/are enabled: *Enable Fully Pipelined Mode, Enable Input Register or Enable Output Register.*
User Interface Ports			
clk_en_i	In	1	Clock enable input. Available if any of the following attributes is/are enabled: *Enable Fully Pipelined Mode, Enable Input Register or Enable Output Register.*
data[n]_i	In	*Width*[1]	Data input.
result_o	Out	*Width*[1]	Output Result.

Note:

1. The bit width of some signals is set by the attribute. Refer to Table 2.2 for the description of these attributes.

2.2. Attribute Summary

The configurable attributes of the Adder Tree Module are shown in Table 2.2 and are described in Table 2.3. The attributes can be configured through the IP Catalog's Module/IP wizard of the Lattice Radiant software.

Table 2.2. Attributes Table

Attribute	Selectable Values	Default	Dependency on Other Attributes
Configuration			
Width	2 – 54	8	—
Number of inputs	2 – N	6	If *Width* >= 2 and *Width* <= 18 and *Enable Fully Pipelined Mode* == Checked, N = 280 else N = 360. If *Width* >= 19 and *Width* <= 36, N = 180. If *Width* >= 37 and *Width* <= 54, N = 120.
Reset Mode	Sync, Async	Sync	Configurable if: *Enable Fully Pipelined Mode* == Checked or *Enable Input Register* == Checked or *Enable Output Register* == Checked
Enable Fully Pipelined Mode	Checked, Unchecked	Unchecked	—
Enable Input Register	Checked, Unchecked	Checked	—
Enable Output Register	Checked, Unchecked	Unchecked	Configurable if *Enable Fully Pipelined Mode* == Unchecked

Table 2.3. Attributes Descriptions

Attribute	Description
Configuration	
Width	Specifies the width of input and output data.
Number of inputs	Specifies the number of inputs to be added together.
Reset Mode	Specifies the mode of reset to be used. *Reset Mode* can only be configured if any of the following attributes is/are enabled: *Enable Fully Pipelined Mode, Enable Input Register or Enable Output Register.*
Enable Fully Pipelined Mode	Specifies if data pipeline is enabled or not. When this is enabled, *Enable Output Register* attribute is automatically enabled.
Enable Input Register	Specifies if input register is enabled or not.
Enable Output Register	Specifies if output register is enabled or not.

3. IP Generation, Simulation, and Validation

This section provides information on how to generate the module using the Lattice Radiant software and how to run simulation and synthesis. For more details on the Lattice Radiant software, refer to the Lattice Radiant software user guide.

3.1. Generating the IP

The Lattice Radiant software allows you to customize and generate modules and IPs and integrate them into the device's architecture. The procedure for generating the Adder Tree Module in the Lattice Radiant software is described below.

To generate the Adder Tree Module:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the IP Catalog tab, double-click Adder_Tree under Module, DSP_Arithmetic_Modules category.
3. The **Module/IP Block Wizard** opens as shown in Figure 3.1. Enter values in the **Component name** and the **Create in** fields and click **Next**.

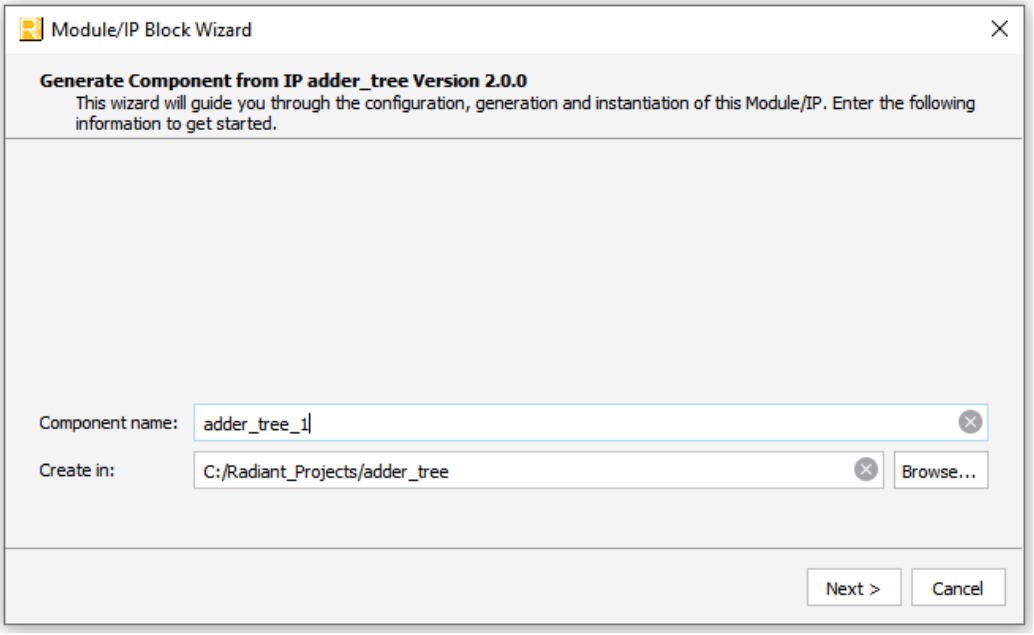

Figure 3.1. Module/IP Block Wizard

4. In the module's dialog box, customize the selected Adder Tree module. A sample configuration is shown in Figure 3.2. For configuration options, see the Attribute Summary section.

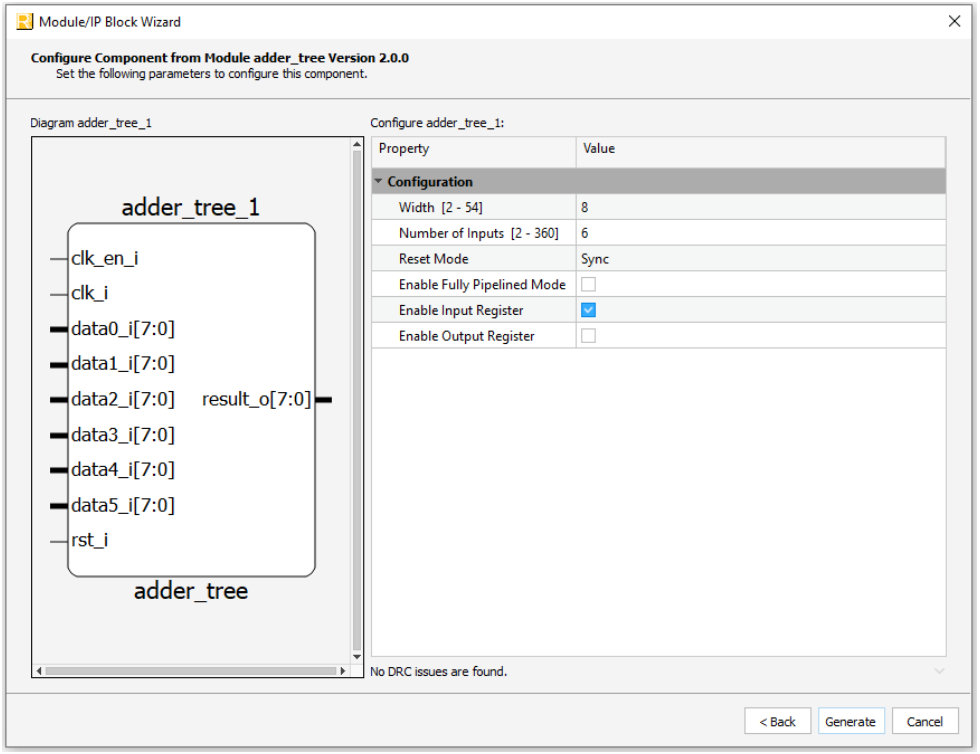

Figure 3.2. Configure User Interface of Adder Tree Module

5. Click **Generate**. The **Check Generating Result** dialog box opens. Design block messages and results are shown in Figure 3.3.

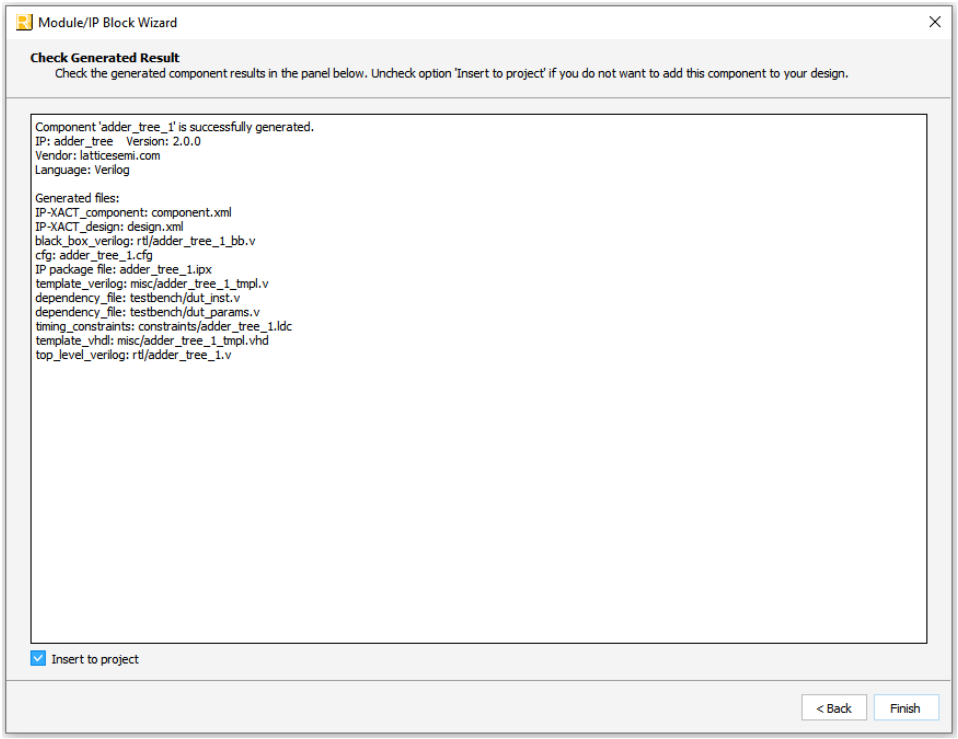

Figure 3.3. Check Generating Result

6. Click **Finish**. All the generated files are placed under the directory paths in the **Create in** and the **Instance name** fields shown in Figure 3.1.

The generated Adder Tree Module package includes the black box (<Instance Name>_bb.v) and instance templates (<Instance Name>_tmpl.v/vhd) that can be used to instantiate the module in a top-level design. An example RTL top-level reference source file (<Instance Name>.v) that can be used as an instantiation template for the module is also provided. You can also use this top-level reference as the starting template for the top-level for their complete design. The generated files are listed in Table 3.1.

Table 3.1. Generated File List

Attribute	Description
<Instance Name>.ipx	This file contains the information on the files associated to the generated IP.
<Instance Name>.cfg	This file contains the attribute values used in IP configuration.
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>.v	This file provides an example RTL top file that instantiates the module.
rtl/<Instance Name>_bb.v	This file provides the synthesis black box.
misc/<Instance Name>_tmpl.v misc /<Instance Name>_tmpl.vhd	These files provide instance templates for the module.
eval/constraints.pdc	IP constraint file.
testbench/tb_top.v	This is the test bench template. You can edit this to match the specific needs.
testbench/dut_params.v	Instantiated version of the <IP_name>.v file for simulation use
testbench/dut_ints.v	Top level parameters of the generated RTL file

3.2. Running Functional Simulation

After the IP is generated, running functional simulation can be performed using different available simulators. The default simulator already has pre-compiled libraries ready for simulation. Choosing a non-default simulator may require additional steps.

To run functional simulation using the default simulator:

1. Click the  button located on the **Toolbar** to initiate the **Simulation Wizard** shown in Figure 3.4.

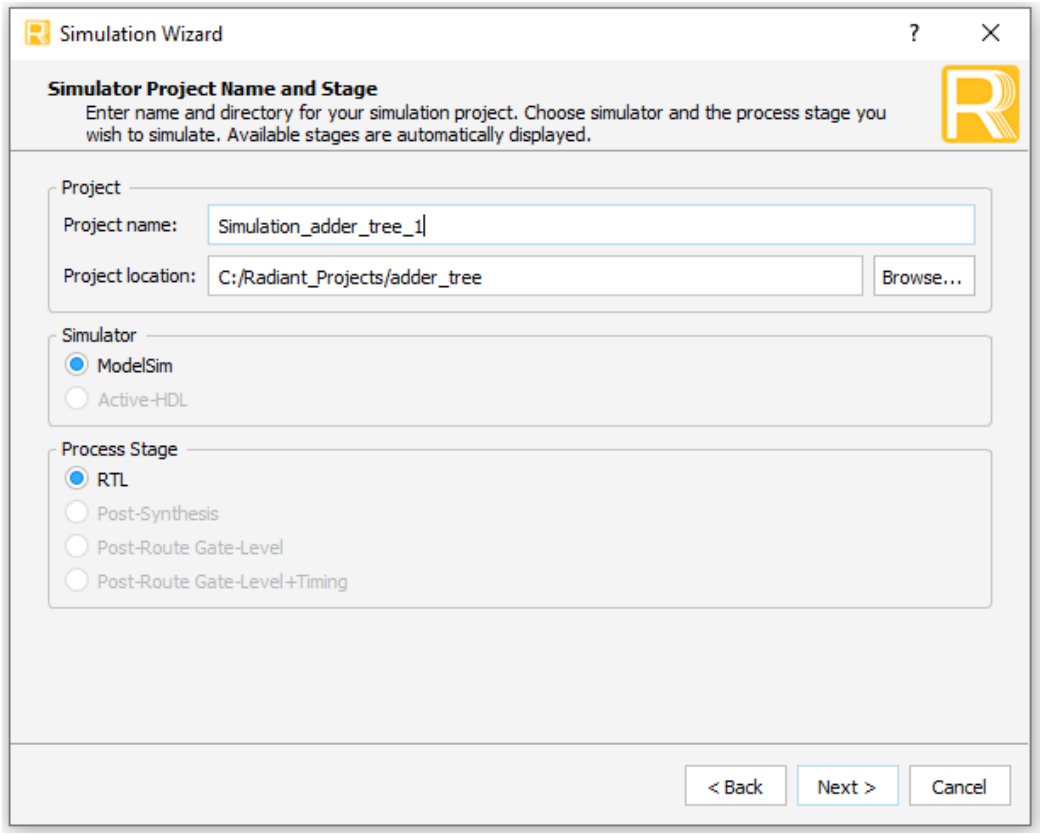

Figure 3.4. Simulation Wizard

2. Click **Next** to open the Add and Reorder Source window as shown in Figure 3.5.

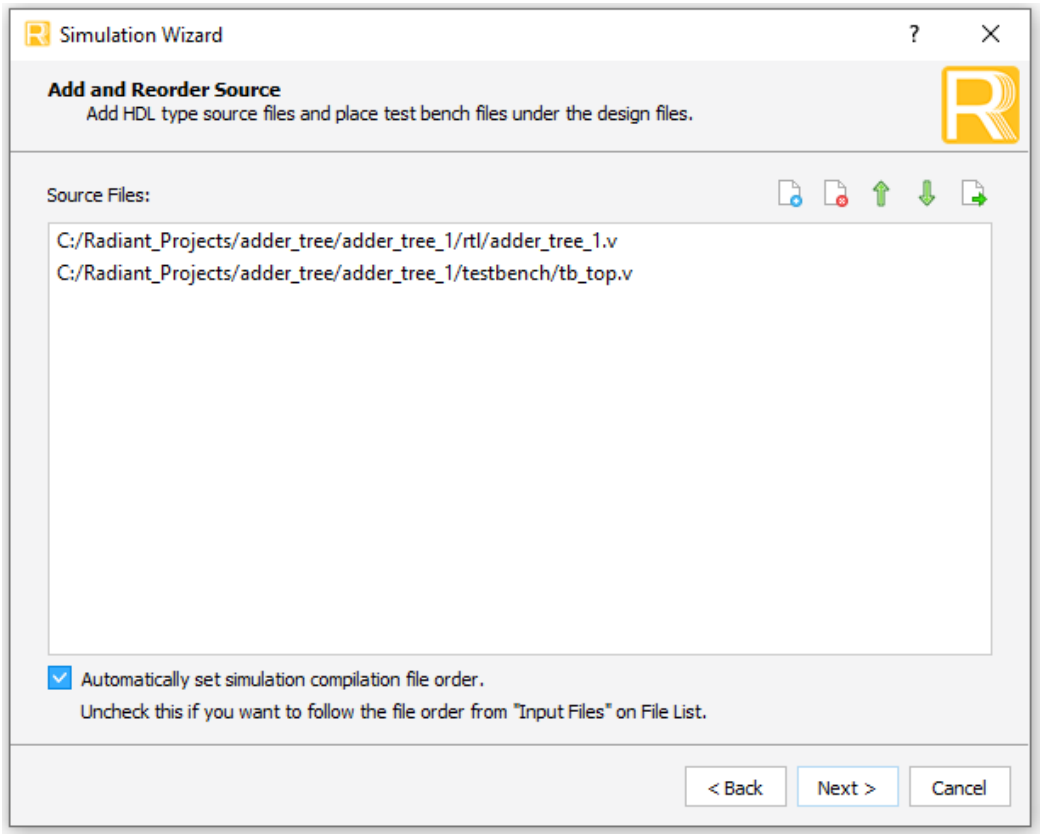

Figure 3.5. Adding and Reordering Source

3. Click **Next**. The **Summary** window is shown.
4. Click **Finish** to run the simulation.

Note: It is necessary to follow the procedure above until it is fully automated in the Lattice Radiant software Suite.

The results of the simulation in our example are provided in Figure 3.6.

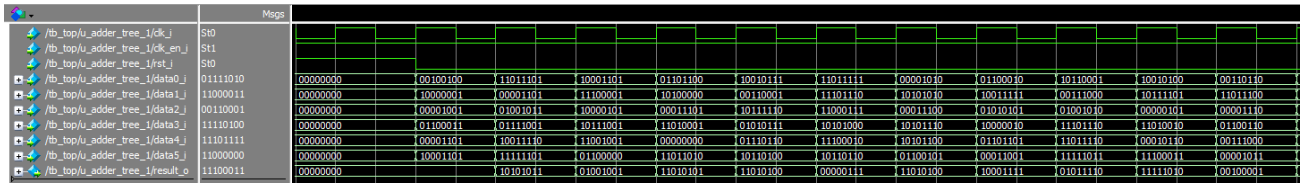

Figure 3.6. Simulation Waveform

3.3. 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:

<IP_Instance_Path>/<IP_Instance_Name>/eval/constraints.pdc.

The constraint file has been verified during IP evaluation with the IP instantiated directly in 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 content of *constraints.pdc* to the top-level design constraint for post-synthesis. Refer to Lattice Radiant Timing Constraints Methodology for details on how to constrain your design.

3.4. IP Evaluation

There is no restriction on the IP evaluation of this module.

Appendix A. Resource Utilization

Table A.1 and Table A.2 show the resource utilization of the Adder Tree Module for LAV-AT-E70-3LFG1156I and LAV-AT-E70-1LFG1156I devices, using Synplify Pro of Lattice Radiant software. Default configuration is used, and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.1. Resource Utilization using LAV-AT-E70-3LFG1156I

Configuration	Clk Fmax (MHz)[1]	Registers	LUTs	EBRs	DSPs
Default	250	0	0	0	3
Number of Inputs = 5, *Reset Mode* = Async, others are default	250	0	0	0	3
Enable Fully Pipelined Mode = Checked, *Width* = 10, others are default	250	60	0	0	3
Enable Input Register = Unchecked, *Enable Output Register* = Unchecked, *Width* = 10, others are default	N/A	0	0	0	3

Note:

1. Fmax is generated when the FPGA design only contains the Adder Tree module and the target frequency is 200 MHz. The obtained Fmax values may be reduced when user logic is added to the FPGA design.

Table A.2. Resource Utilization using LAV-AT-E70-1LFG1156I

Configuration	Clk Fmax (MHz)[1]	Registers	LUTs	EBRs	DSPs
Default	250	0	0	0	3
Number of Inputs = 5, *Reset Mode* = Async, others are default	250	0	0	0	3
Enable Fully Pipelined Mode = Checked, *Width* = 10, others are default	250	60	0	0	3
Enable Input Register = Unchecked, *Enable Output Register* = Unchecked, *Width* = 10, others are default	N/A	0	0	0	3

Note:

1. Fmax is generated when the FPGA design only contains the Adder Tree module, and the target frequency is 200 MHz. The obtained Fmax values may be reduced when user logic is added to the FPGA design.

References

For more information refer to:

- Avant-E web page
- Avant-G web page
- Avant-X web page
- IP Cores for Avant devices
- Lattice Radiant Timing Constraints Methodology
- Lattice Radiant FPGA design software
- Lattice Radiant software user guide
- Lattice Insights for Lattice Semiconductor training courses 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 1.2, December 2023

Section	Change Summary
All	Performed minor editorial fixes.
Disclaimers	Updated this section.
IP Generation, Simulation, and Validation	• Updated Table 3.1. • Added section 3.3 Constraining the IP.
Resource Utilization	• Updated device names from *LAV-AT-500E-* to *LAV-AT-E70-*. • Updated the Clk Fmax values in Table A.1 and Table A.2.
References	Added relevant links.

Revision 1.1, March 2023

Section	Change Summary
All	Updated document template and format fixes.
Technical Support Assistance	Added Lattice Answer Database website link.

Revision 1.0, October 2022

Section	Change Summary
Functional Description	Added below figures: • Figure 2.2. Adder Tree Enabled Input Register Implementation Diagram • Figure 2.3. Adder Tree Enabled Output Register Implementation Diagram • Figure 2.4. Adder Tree Enabled Input and Output Register Implementation Diagram • Figure 2.5. Adder Tree Enabled Output Register and Fully Pipelined Mode Implementation Diagram • Figure 2.6. Adder Tree Enabled Input and Output Register and Fully Pipelined Mode Implementation Diagram
IP Generation, Simulation, and Validation	• Removed “Licensing the IP” section. • Updated title of the Section 3.1 from “Generating and Synthesizing” to “Generating the IP”. • Added Figure 3.2. Configure User Interface of Adder Tree Module.
Appendix A. Resource Utilization	Added Resource Utilization section.

Revision 0.80, May 2022

Section	Change Summary
All	Initial release.

www.latticesemi.com