AHB-Lite Manager BFM Lite VIP – Lattice Propel Builder 2026.1

IP Version: v1.0.2

User Guide

FPGA-IPUG-02328-1.0

June 2026

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 REDUNDANCIES, 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.

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
VIP	Verification Intellectual Property
AHB-Lite	Advanced High-performance Bus – Lite
API	Application Programming Interface
BFM	Bus Functional Model
DUT	Design Under Test
FPGA	Field Programmable Gate Array
RISC-V	Reduced Instruction Set Computer-V (Five)
RISC-V MC	RISC-V for Microcontroller Applications
SoC	System-on-Chip

1. Introduction

The Lattice Semiconductor AHB-Lite Manager Bus Functional Model (BFM) Lite Verification IP (VIP) is for verification purposes only. It generates different types of AHB-Lite transactions, including single transactions, pipelined transactions, and burst transactions.

The AHB-Lite Manager BFM Lite VIP design is implemented in SystemVerilog. It can be configured and generated using the Lattice Propel™ Builder software. It is for simulation only and is not targeted to any device.

1.1. What's New in This IP Release

Added logs for every task to clarify the read and write results.

1.2. Features

The AHB-Lite Manager BFM Lite VIP has the following features:

- AHB-Lite manager interface
- Single read and write transactions generation
- Pipelined read and write transactions generation
- Burst read and write transactions generation

1.3. Conventions

1.3.1. Nomenclature

The nomenclature used in this document is based on SystemVerilog.

1.3.2. Signal Names

- _n are active low signals, asserted when the value is logic 0.
- _i are input signals.
- _o are output signals.
- _io are bidirectional signals.

2. Functional Descriptions

2.1. Overview

The AHB-Lite Manager BFM Lite VIP is used to generate AHB-Lite transactions for the verification environment, as shown in Figure 2.1. Supported transactions include single read and write transactions, pipelined read and write transactions, and burst read and write transactions.

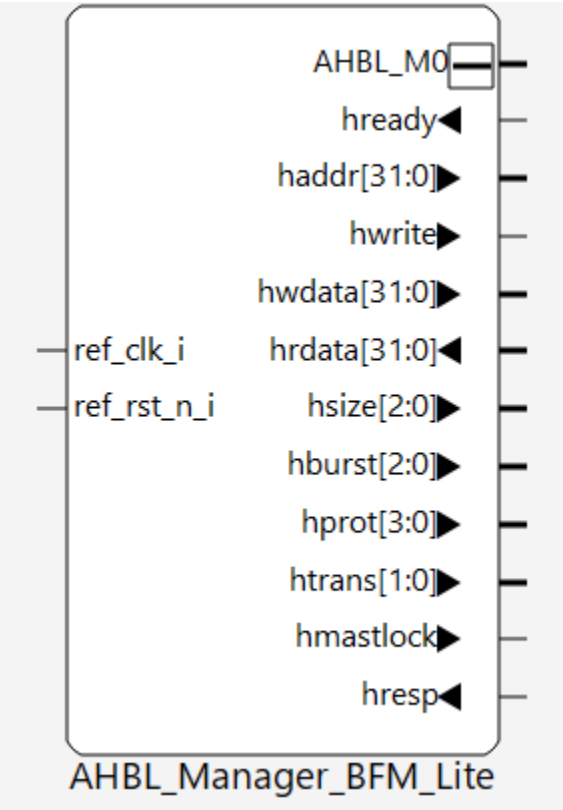

Figure 2.1. AHB-Lite Manager BFM Lite Block Diagram

2.2. Signal Description

Table 2.1 and Table 2.2

Table 2.2 list the ports of the AHB-Lite Manager BFM Lite VIP.

Table 2.1. Clock and Reset Port

Name	Type	Width	Description
ref_clk_i	In	1	Clock input
ref_rst_n_i	In	1	Reset input, active low

Table 2.2. AHB-Lite Manager Port

Name	Type	Width	Description
HREADY	In	1	Standard AHB-Lite Manager interface
HRESP	In	1	Standard AHB-Lite Manager interface
HRDATA	In	32	Standard AHB-Lite Manager interface
HWRITE	Out	1	Standard AHB-Lite Manager interface
HTRANS	Out	3	Standard AHB-Lite Manager interface

Name	Type	Width	Description
HSIZE	Out	3	Standard AHB-Lite Manager interface
HBURST	Out	3	Standard AHB-Lite Manager interface
HPROT	Out	4	Standard AHB-Lite Manager interface
HMASTLOCK	Out	1	Standard AHB-Lite Manager interface
HADDR	Out	32	Standard AHB-Lite Manager interface
HWDATA	Out	32	Standard AHB-Lite Manager interface

Refer to AMBA 3 AHB-Lite Protocol v1.0 for more information.

2.3. Attribute Summary

The configurable attributes of the AHB-Lite Manager BFM Lite VIP are shown in Table 2.3. The attributes can be configured through the Propel Builder software.

Table 2.3. Configurable Attributes

Attribute	Selectable Values	Default	Description
ASSERTION_EN	0, 1	0	Internal assertions enable. 0 – Disable. 1 – Enable. * Reserved for future use.

2.4. API Summary

Table 2.4. API Summary

API	Description
write_word_single (input logic [31:0] addr, input logic [31:0] wdata)	Generate a single word write transaction. “addr” is the target address, and “wdata” is the word to be written.
write_word_pipeline (input logic [31:0] addr[], input logic [31:0] wdata[])	Generate pipelined word write transactions. “addr[]” contains the target addresses, and “wdata[]” contains the words to be written.
write_word_burst_incr (input logic [31:0] base_addr, input int beats, input logic [31:0] wdata[])	Generate an incremental burst write transaction. “base_addr” is the start address, “beats” is the burst beats, and “wdata[]” contains the words to be written.
read_word_single (input logic [31:0] addr, output logic [31:0] rdata)	Generate a single word read transaction. “addr” is the target address, and “rdata” is connected to HRDATA and reflects the read data.
read_word_pipeline (input logic [31:0] addr[], output logic [31:0] rdata[])	Generate pipelined word read transactions. “addr[]” contains the target addresses, and “rdata[]” is connected to HRDATA and reflects the read data.
read_word_burst_incr (input logic [31:0] base_addr, input int beats, output logic [31:0] rdata[])	Generate an incremental burst read transaction. “base_addr” is the start address, “beats” is the burst beats, and “rdata[]” is connected to HRDATA and reflects the read data.

The following is an example of API usage.

```
logic [31:0] waddr_word_single = 32'h00000010;
 logic [31:0] wdata_word_single = 32'h10101010;
logic [31:0] raddr_word_single = 32'h00000010;
 logic [31:0] rdata_word_single;
logic [31:0] waddr_word_pipeline[3] = '{32'h00000014, 32'h00000018, 32'h0000001C};
 logic [31:0] wdata_word_pipeline[3] = '{32'h20202020, 32'h30303030, 32'h40404040};
logic [31:0] raddr_word_pipeline[3] = '{32'h00000010, 32'h00000014, 32'h00000018};
 int rnums_word_pipeline = 3;
 logic [31:0] rdata_word_pipeline[3];
logic [31:0] waddr_word_burst_incr = 32'h00000010;
 int wbeats_word_burst_incr = 5;
 logic [31:0] wdata_word_burst_incr[5] = '{32'h00001001, 32'h00001002, 32'h00001003, 32'h00001004,
32'h00001005};
logic [31:0] raddr_word_burst_incr = 32'h00000010;
 int rbeats_word_burst_incr = 5;
 logic [31:0] rdata_word_burst_incr[5];
initial begin
  wait(ahbl_manager_example_inst_ref_rst_n_i_net);
  repeat(5) @(posedge ahbl_manager_example_inst_ref_clk_i_net);
  // Single write
  ahbl_manager_example_inst.write_word_single(waddr_word_single, wdata_word_single);
  // Single read
  ahbl_manager_example_inst.read_word_single(raddr_word_single, rdata_word_single);
  // Pipelined write
  ahbl_manager_example_inst.write_word_pipeline(waddr_word_pipeline, wdata_word_pipeline);
  // Pipelined read
  ahbl_manager_example_inst.read_word_pipeline(raddr_word_pipeline, rnums_word_pipeline,
rdata_word_pipeline);
  // INCR burst write
  ahbl_manager_example_inst.write_word_burst_incr(waddr_word_burst_incr, wbeats_word_burst_incr,
wdata_word_burst_incr);
  // INCR burst read
  ahbl_manager_example_inst.read_word_burst_incr(raddr_word_burst_incr, rbeats_word_burst_incr,
rdata_word_burst_incr);
 end
```

3. AHB-Lite Manager BFM Lite VIP Generation

This section provides information on how to generate the AHB-Lite Manager BFM Lite VIP module using the Lattice Propel Builder software.

To generate the AHB-Lite Manager BFM Lite VIP module:

1. In Lattice Propel Builder, create a new Scalable RISC-V SoC Project and select the RISC-V MC core (Figure 3.1).

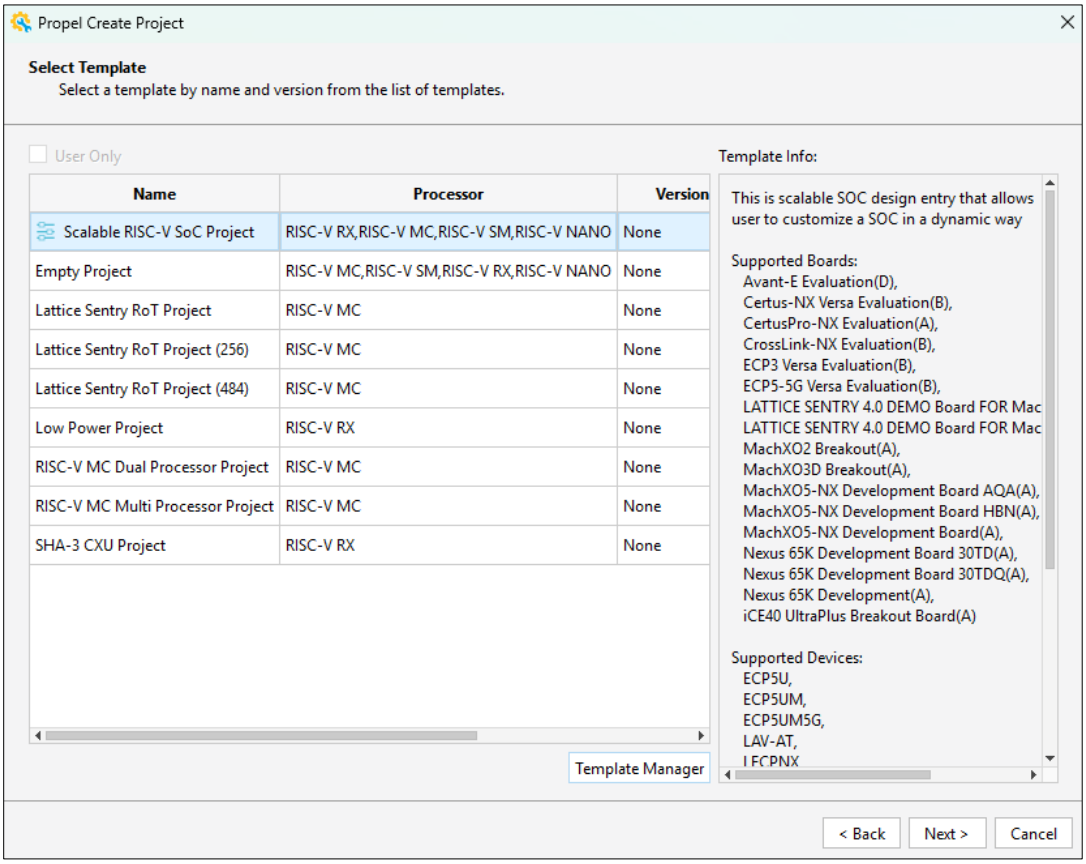

Figure 3.1. Scalable RISC-V SoC Project

2. Modify the AHB-Lite Interconnect instance and increase Total AHB-Lite Managers by one (Figure 3.2).

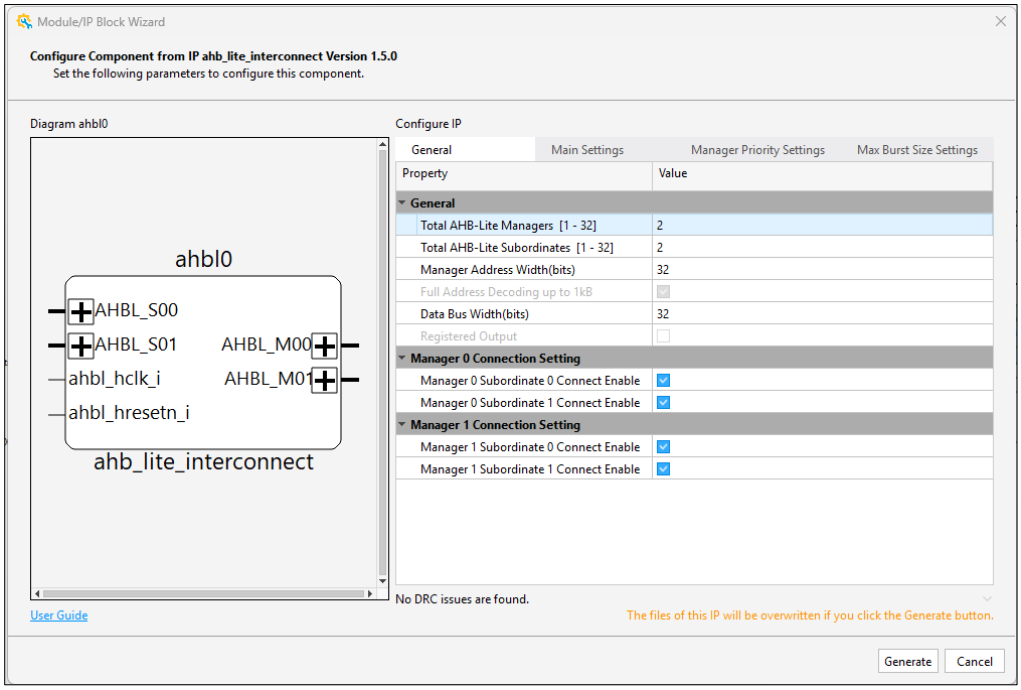

Figure 3.2. AHB-Lite Interconnect

3. Add the AHB-Lite Feedthrough IP into the design and export the interface as Manager (Figure 3.3).

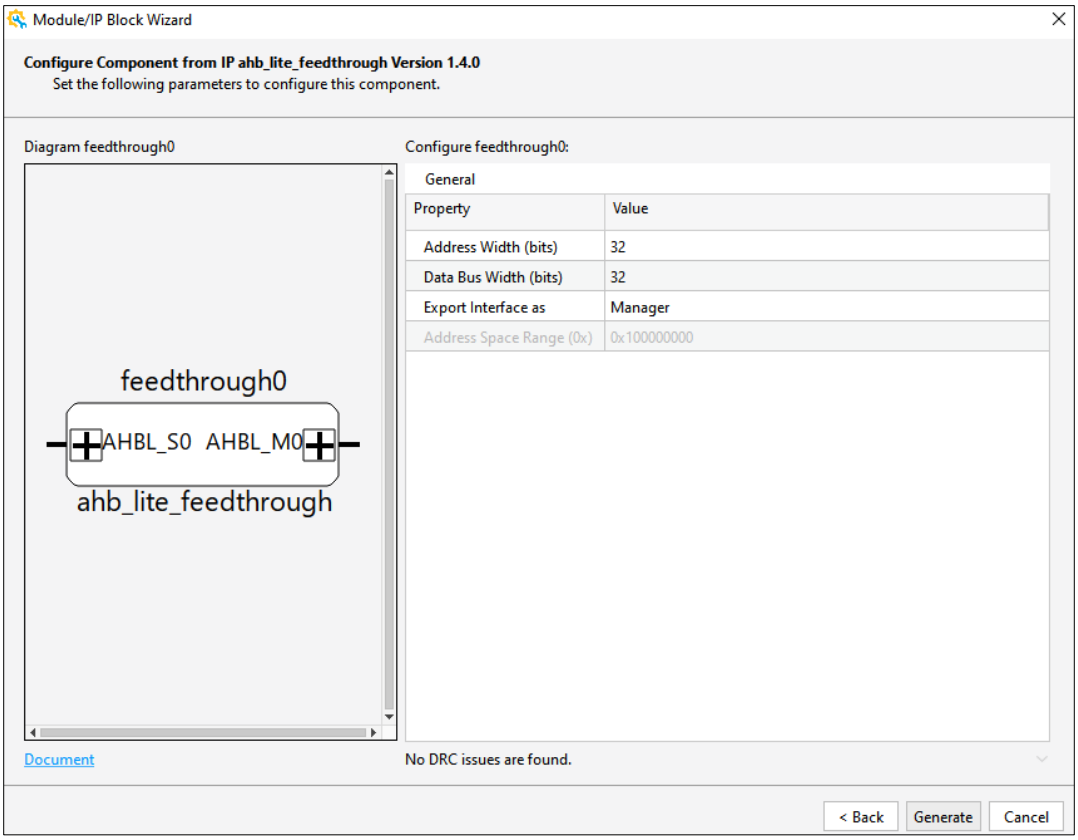

Figure 3.3. AHB-Lite Feedthrough

4. Connect the AHB-Lite Feedthrough instance to AHB-Lite Interconnect and export the interface to top (Figure 3.4).

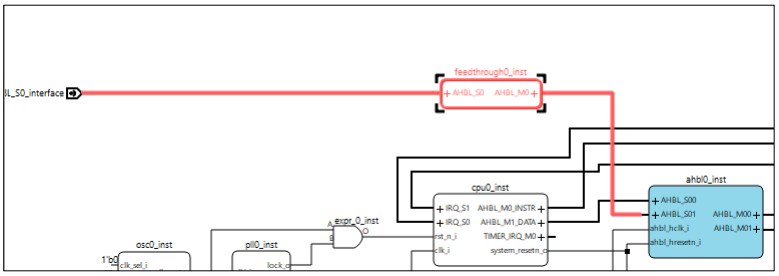

Figure 3.4. Export of Feedthrough

5. Export the clock and reset to top (Figure 3.5).

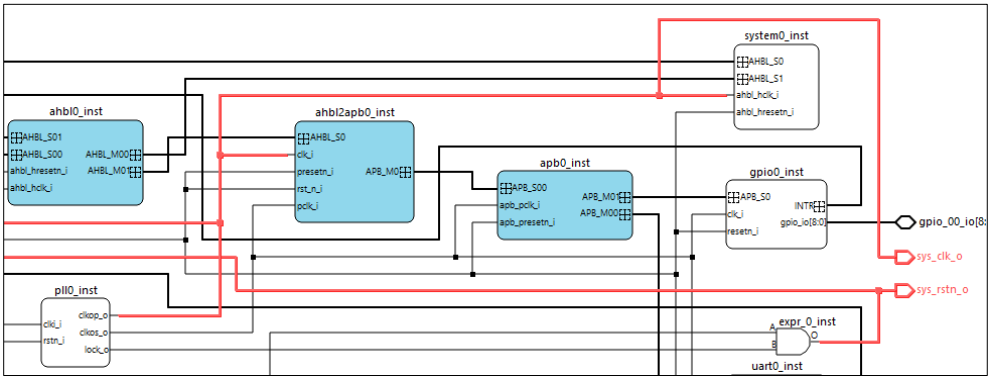

Figure 3.5. Export of Clock and Reset

6. Select **Design** > **Switch Verification and SoC Design** (Figure 3.6).

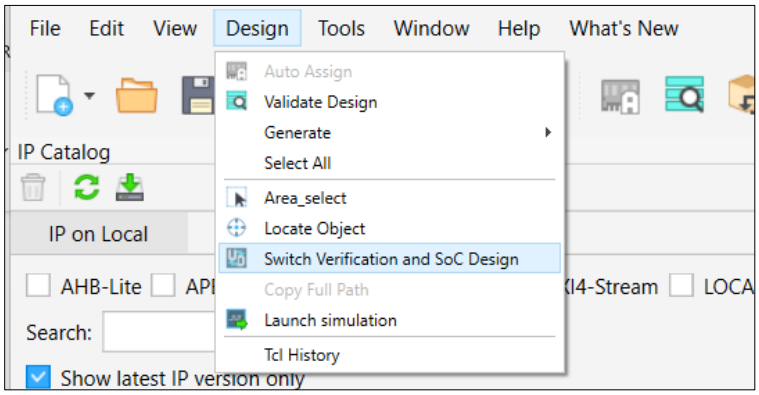

Figure 3.6. Switch Verification and SoC Design

7. Select the AHB-Lite Manager BFM Lite VIP from IP Catalog (Figure 3.7).

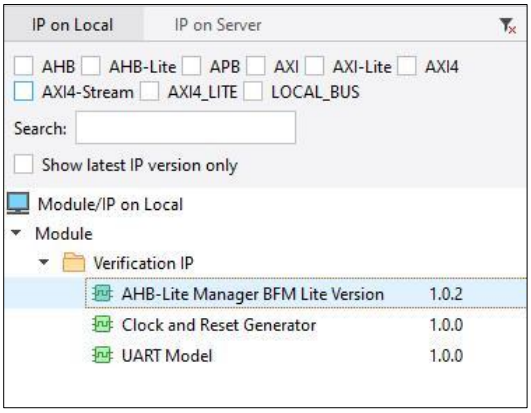

Figure 3.7. AHB-Lite Manager BFM Lite VIP

8. Enter the component name. Click **Next** (Figure 3.8).

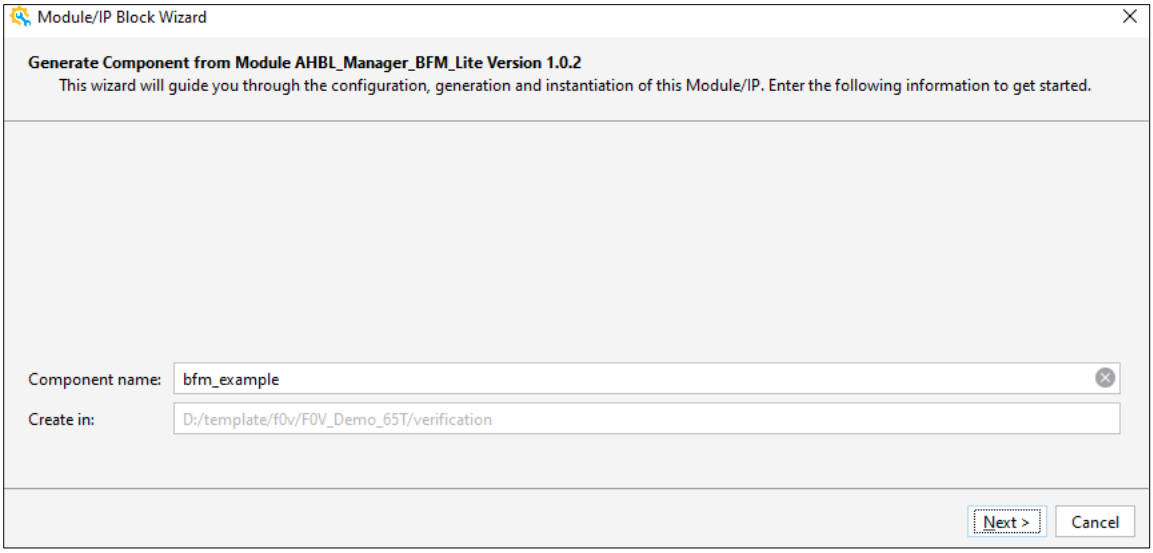

Figure 3.8. Entering Component Name

9. Configure the parameters as needed. Click **Generate** (Figure 3.9).

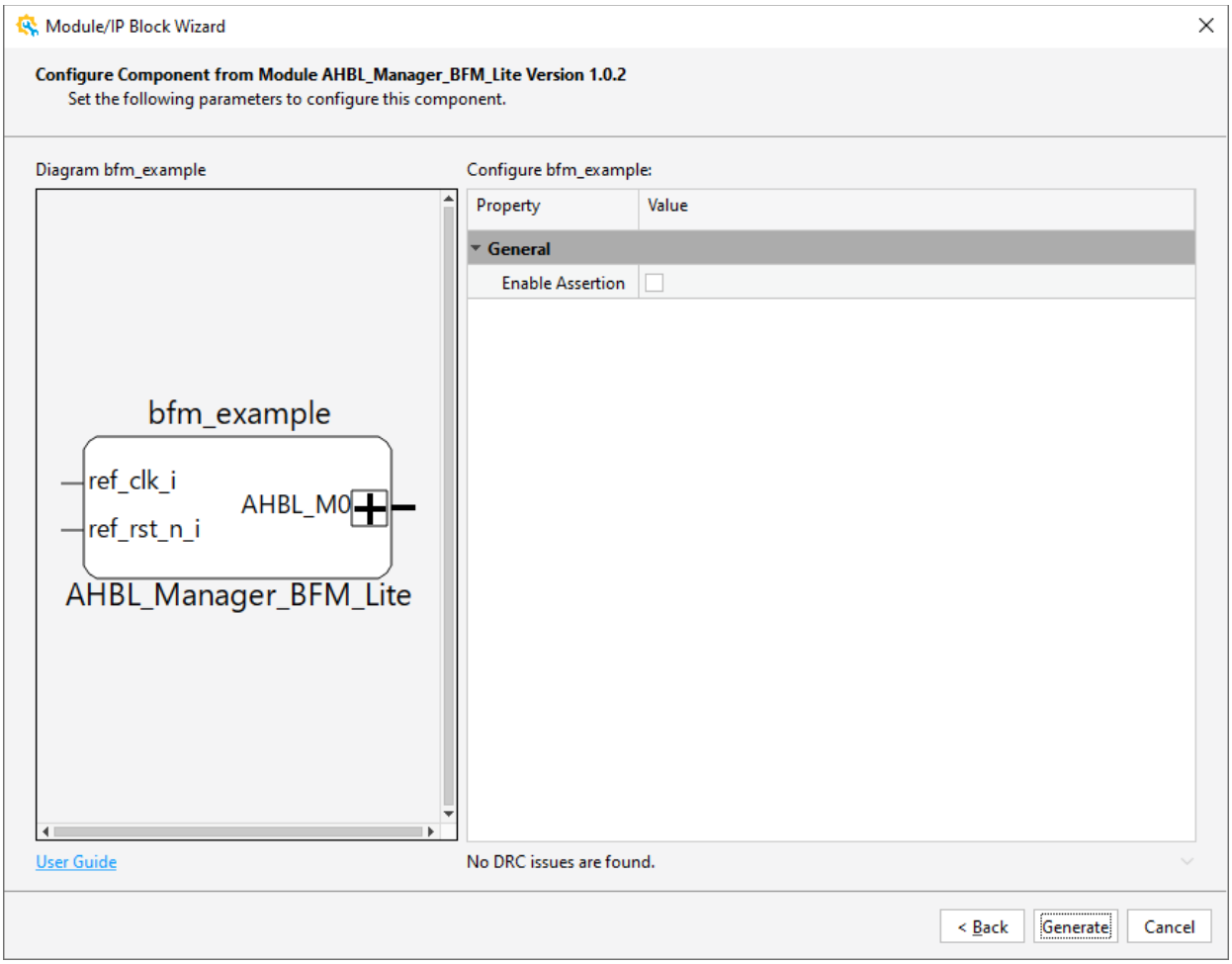

Figure 3.9. Configuring Parameters

10. Verify the information. Click **Finish** (Figure 3.10).

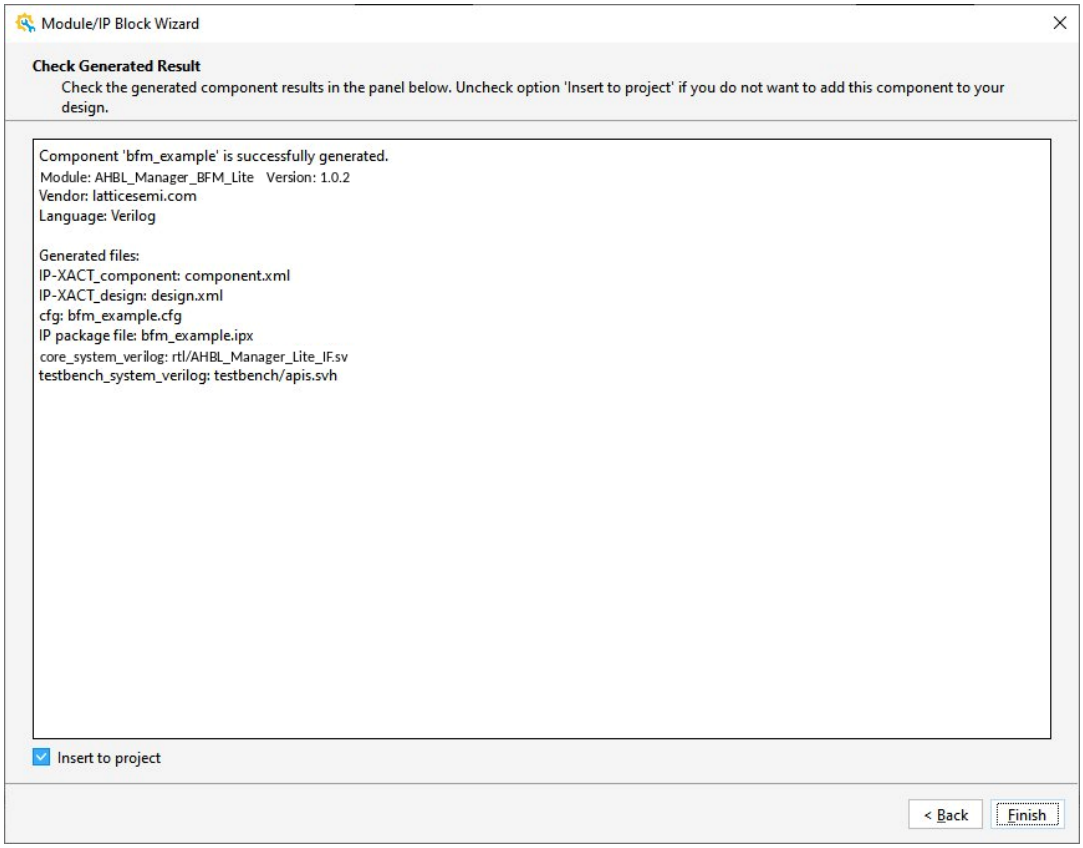

Figure 3.10. Verifying Result

11. Confirm or modify the module instance name. Click **OK** (Figure 3.11).

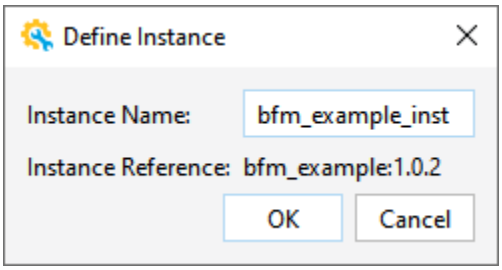

Figure 3.11. Specifying Instance Name

12. The AHB-Lite Manager BFM Lite VIP instance is successfully generated (Figure 3.12).

Figure 3.12. Generated Instance

13. Connect the AHB-Lite Manager BFM Lite VIP instance to the design under test (DUT) instance (Figure 3.13).

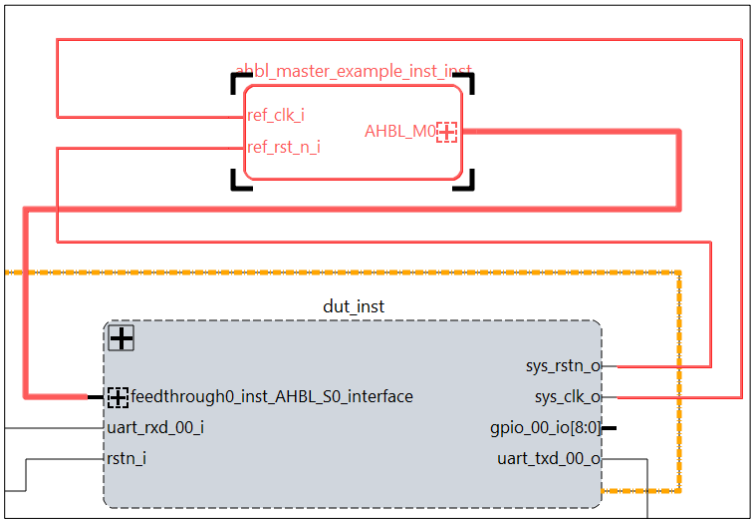

Figure 3.13. Connect BFM Instance

14. Click the **Generate** button to generate the testbench (Figure 3.14). The Simulation Wizard window pops up. Set the run time for simulation and click **Generate**.

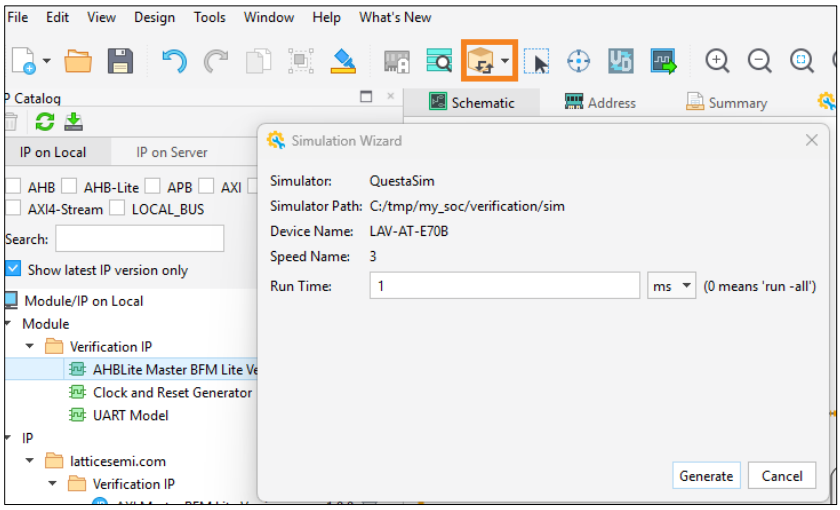

Figure 3.14. Generate Testbench

15. In a text editor, open the testbench source file, for example,
`<your design path>\<design>\verification\sim\<design>_v.sv`.
Copy the example code of API usage from the API Summary section into the selected area shown in Figure 3.15.

```
123
124     /*----------------------------BFM Block----------------------------*/
125     /* This section is reserved for user to create stimulus using BFM */
126     /* APIs.                                                          */
127     /*-----------------------------------------------------------------*/
128
129
130     initial begin
131         //Enabled Predefined sequences
132
133         //Reserved for User to write BFMs
134
135     end
136
137     initial begin
138       axi_mst_inst.set_assertion(0);
139     end
140
```

Figure 3.15. Testbench Source File

16. Re-run the simulation using either of the following options (Figure 3.16).
    - Close the simulator and relaunch it.
    - Type `quit -sim` then `do qsim.do` in the QuestaSim transcript area.

```
# [1726000] AXI BFM: read_word_single A:0x30000 D:0x1
VSIM 4> quit -sim
# End time: 17:08:09 on Mar 26,2026, Elapsed time: 0:00:54
# Errors: 0, Warnings: 10

Questa Lattice OEM> do qsim.do
```

Figure 3.16. Run Simulation

17. The transaction is displayed (Figure 3.17).

```
#    Time: 0 ps  Iteration: 0  Instance: /my_soc_ahbl_v/sysclk_rst_gen1_inst File: my_soc_ahbl_v.sv Line: 205
# Testbench clock frequency set to 50.000000 Mhz, clock period set to 20.000000 ns
# [63112000] BFM: write_word_single = A:0x10, D:0x10101010
# [63181000] BFM: read_word_single = A:0x10, D:0x10101010
# [63191000] BFM: write_word_pipeline = A:0x14, D:0x20202020
# [63201000] BFM: write_word_pipeline = A:0x18, D:0x30303030
# [63211000] BFM: write_word_pipeline = A:0x1c, D:0x40404040
# [63286000] BFM: read_word_pipeline = A:0x10, D:0x10101010
# [63326000] BFM: read_word_pipeline = A:0x14, D:0x20202020
# [63366000] BFM: read_word_pipeline = A:0x18, D:0x30303030
# [63381000] BFM: write_word_burst_incr = Beats:0x0, A:0x10, D:0x1001
# [63391000] BFM: write_word_burst_incr = Beats:0x1, A:0x14, D:0x1002
# [63401000] BFM: write_word_burst_incr = Beats:0x2, A:0x18, D:0x1003
# [63411000] BFM: write_word_burst_incr = Beats:0x3, A:0x1c, D:0x1004
# [63421000] BFM: write_word_burst_incr = Beats:0x4, A:0x20, D:0x1005
# [63496000] BFM: read_word_burst_incr = Beats:0x0, A:0x10, D:0x1001
# [63506000] BFM: read_word_burst_incr = Beats:0x1, A:0x14, D:0x1002
# [63516000] BFM: read_word_burst_incr = Beats:0x2, A:0x18, D:0x1003
# [63526000] BFM: read_word_burst_incr = Beats:0x3, A:0x1c, D:0x1004
# [63536000] BFM: read_word_burst_incr = Beats:0x4, A:0x20, D:0x1005

VSIM 3>
```

Figure 3.17. Simulation Output

References

- Lattice Propel Builder 2026.1 User Guide (FPGA-UG-02254)
- AMBA 3 AHB-Lite Protocol v1.0
- Lattice Propel™ design environment web page
- Lattice Radiant™ software 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 1.0, IP v1.0.2, June 2026

Section	Change Summary
All	Production release.

www.latticesemi.com