# AHBL Master BFM Lite VIP - Lattice Propel Builder

# User Guide

FPGA-IPUG-02148-1.0

November 2020

## 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 and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. 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. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. 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 |
|---|---|
| VIP | Verification Intellectual Property |
| AHBL | Advanced High-performance Bus - Lite |
| FPGA | Field Programmable Gate Array |
| HDL | Hardware Description Language |
| LUT | Lookup-Table |

# 1. Introduction

The Lattice Semiconductor AHBL Master BFM Lite Verification IP (VIP) is for verification purpose only. It is able to generate different types of AHBL transactions, including single transactions, pipelined transactions and burst transactions.

The AHBL Master BFM Lite VIP design is implemented in System Verilog. 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. Features

The AHB-Lite master has the following features:

- AHB-Lite master interface.
- Single read/write transactions generation.
- Pipelined read/write transactions generation.
- Burst read/write transactions generation.

## 1.2. Conventions

### 1.2.1. Nomenclature

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

# 2. Functional Descriptions

## 2.1. Overview

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

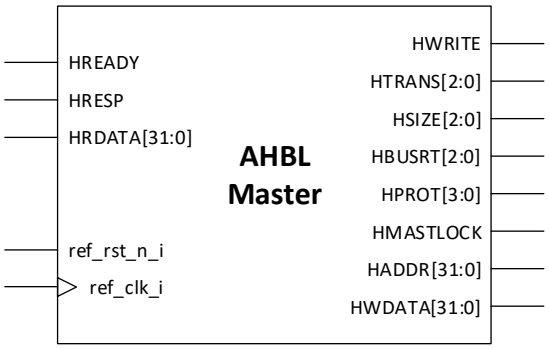


**Figure 2.1. AHBL Master BFM Lite Block Diagram**

## 2.2. Signal Description

Table 2.1 and Table 2.2 list the ports of the AHBL Master 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. AHBL Master Port**

| Name | Type | Width | Description |
|---|---|---|---|
| HREADY | In | 1 | Standard AHB-Lite Slave interface |
| HRESP | in | 1 | Standard AHB-Lite Slave interface |
| HRDATA | in | 32 | Standard AHB-Lite Slave interface |
| HWRITE | out | 1 | Standard AHB-Lite Slave interface |
| HTRANS | out | 3 | Standard AHB-Lite Slave interface |
| HSIZE | out | 3 | Standard AHB-Lite Slave interface |
| HBURST | out | 3 | Standard AHB-Lite Slave interface |
| HPROT | out | 4 | Standard AHB-Lite Slave interface |
| HMASTLOCK | out | 1 | Standard AHB-Lite Slave interface |
| HADDR | out | 32 | Standard AHB-Lite Slave interface |
| HWDATA | out | 32 | Standard AHB-Lite Slave interface |

Refer to AMBA 3 AHB-Lite Protocol V1.0 for more information.

## 2.3. Attribute Summary

The configurable attributes of the AHBL Master 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.<br>0 – disable<br>1 – enable |

## 2.4. API Summary

**Table 2.4. API Summary**

| API | Description |
|---|---|
| write_word_single<br>(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<br>(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<br>(input logic [31:0] base_addr, input int beats, input logic [31:0] wdata[]) | Generate an incremental burst write transaction.”addr” is the start address, “beats” is the burst beats, and “wdata[]” contains the words to be written. |
| read_word_single<br>(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<br>(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<br>(input logic [31:0] base_addr, input int beats, output logic [31:0] rdata) | Generate an incremental burst read transaction.”addr” is the start address, “beats” is the burst beats, and “rdata” is connected to HRDATA and reflects the read data. |

Following is an example of API usage.

```
   logic [31:0]  waddr_word_single;
   logic [31:0]  wdata_word_single;

   logic [31:0]  waddr_word_burst_incr = 32'h00001000;
   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_pipeline[3] = '{32'h00000100, 32'h00000200, 32'h00000300};
   logic [31:0]  rdata_word_pipeline;

   initial begin
     // Single write
      waddr_word_single = 32'h00000010;
      wdata_word_single = 32'h10101010;
      ahbl_master_if.write_word_single(waddr_word_single, wdata_word_single);

      // Pipelined read
      ahbl_master_if.read_word_pipeline(raddr_word_pipeline, rdata_word_pipeline);

      // INCR burst write
      ahbl_master_if.write_word_burst_incr(waddr_word_burst_incr, wbeats_word_burst_incr,
wdata_word_burst_incr);
   end
```

# 3. AHBL Master BFM Lite VIP Generation

This section provides information on how to generate the AHBL Master BFM Lite VIP module using Propel Builder.

To generate the AHBL Master BFM Lite VIP module:

1. In the Propel Builder, create a new verification design, and select the AHBL Master BFM Lite VIP package.
2. Enter the component name. Click **Next** (Figure 3.1).

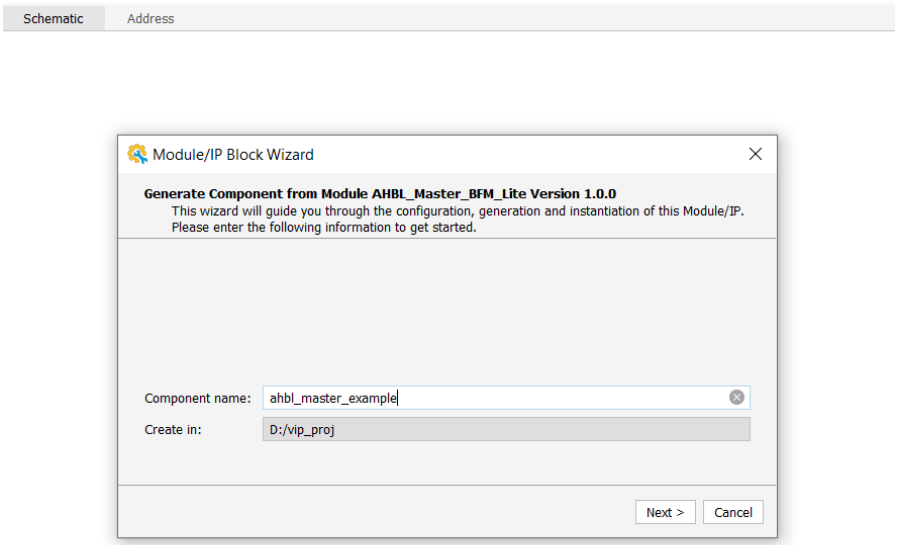


**Figure 3.1. Entering Component Name**

3. Configure the parameters as needed. Click **Generate** (Figure 3.2).

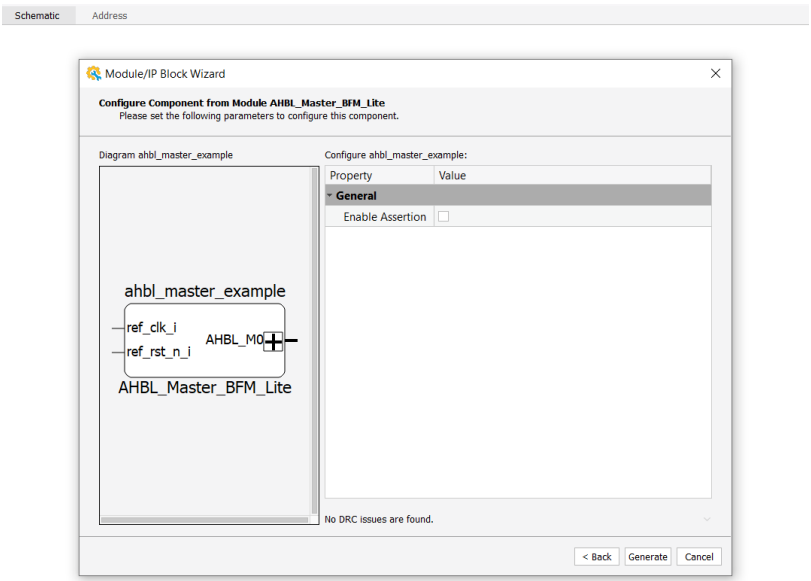


**Figure 3.2. Configuring Parameters**

4. Verify the information. Click **Finish** (Figure 3.3).

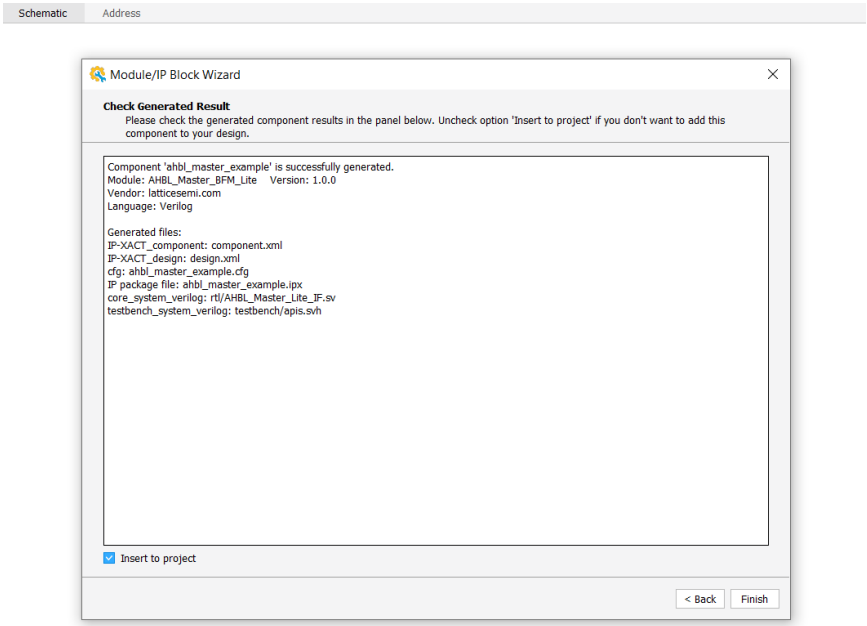


**Figure 3.3. Verifying Result**

5. Confirm or modify the module instance name. Click **OK** (Figure 3.4).

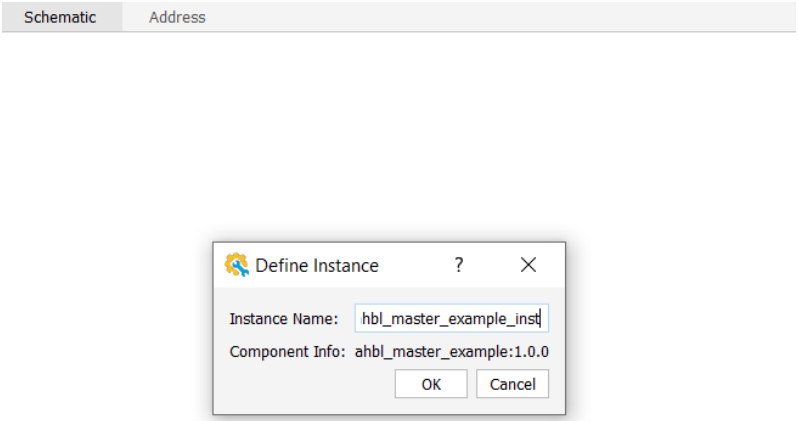


**Figure 3.4. Specifying Instance Name**

The AHBL Master BFM Lite VIP instance is successfully generated (Figure 3.5).

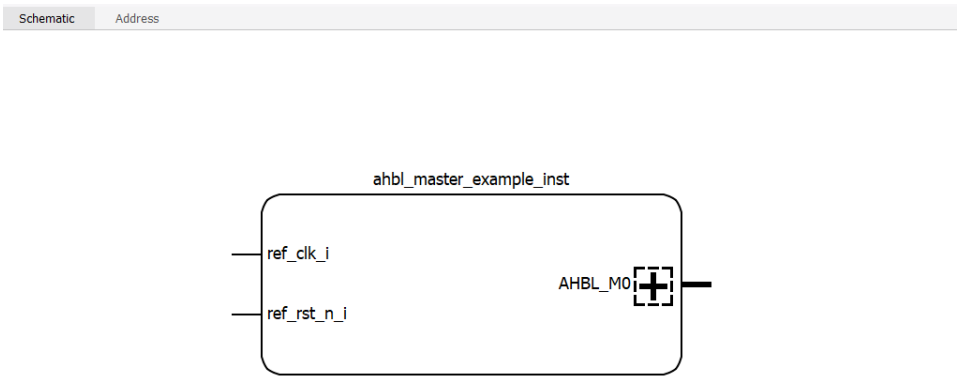


**Figure 3.5. Generated Instance**

# References

- Lattice Semiconductor MachXO3D FPGA Web Page
- Lattice Semiconductor CrossLink-NX FPGA Web Page
- Lattice Propel 1.0 User Guide
- Lattice Diamond Software 3.11 User Guide
- Lattice Radiant Software 2.1 User Guide
- AMBA 3 AHB-Lite Protocol V1.0

# Technical Support Assistance

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

# Revision History

**Revision 1.0, November 2020**

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


www.latticesemi.com