# Using MachXO3D ESB to Implement SHA256

## Reference Design

FPGA-RD-02054-1.0

August 2021

## 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 |
|---|---|
| AES | Advanced Encryption Standard |
| ECDSA | Elliptic Curve Digital Signature Algorithm |
| ESB | Embedded Security Block |
| HSP | High Speed Port |
| SHA256 | 256-bit Secure Hash Algorithm |

# 1. Introduction

A hash function maps an input of arbitrary length into a fixed number of output bits, the digest or hash value. This digest should be the same each time the same input is hashed. The hash function has two properties:

- If someone gets a digest, determining its original script is difficult or impossible.
- Having two different messages that can be hashed to the same digest is difficult or impossible.

SHA256 is a novel hash function with 256 bits fixed output. The most important use of this hash function is the information authentication protection and use as a digital signature schemes tool.

The MachXO3D™ device is the next product family of the MachXO™ product line with key features such as security and on-chip dual boot. The ESB submodule of the MachXO3D device focuses on security features and involves cryptographic functions such as Elliptic Curve Digital Signature Algorithm (ECDSA), Advanced Encryption Standard (AES), and SHA256. This reference design takes advantage of the ESB implementing SHA256 function.

**Note**: Knowledge of the SHA256 algorithm details is not a requirement.

# 2. Reference Design Overview

## 2.1. Block Diagram

Figure 2.1 shows the block diagram of the reference design using the WISHBONE bus.

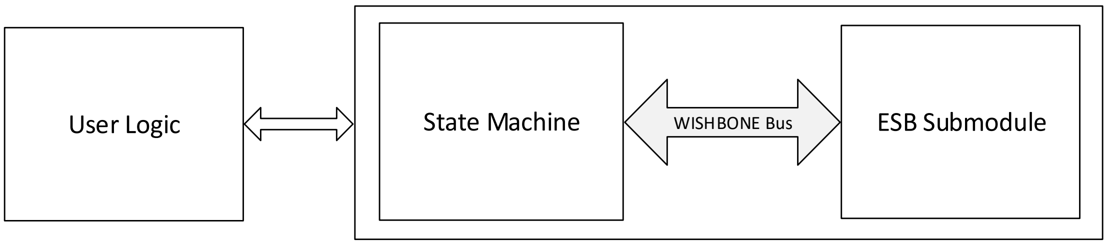


**Figure 2.1. Top-Level Block Diagram**

Figure 2.2 shows the block diagram of the reference design using the WISHBONE bus and the HSP bus.

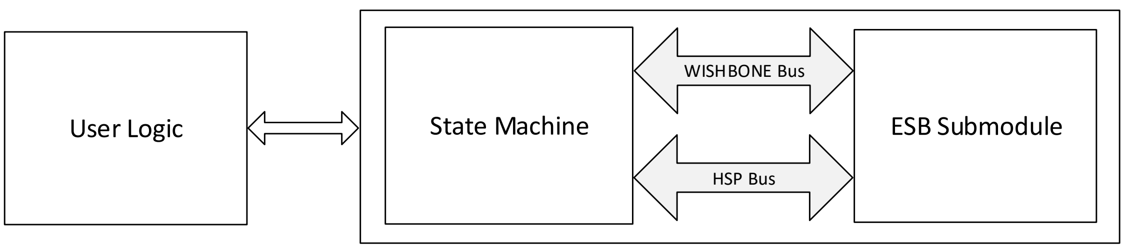


**Figure 2.2. Top-Level Block Diagram**

## 2.2. Overview

This design provides two ways to access the ESB. One way is to use the WISHBONE interface to access the register of the ESB and transmit data between the state machine and the ESB, as shown in Figure 2.1. The other way is to use the WISHBONE interface to access the register of the ESB and to use HSP interface to transmit data between the state machine and the ESB, as shown in Figure 2.2. The state machine shows the detailed steps of accessing the register and the ESB memory. The user logic can utilize the state machine for SHA256 implementation. The features include:

- Simplify the user design and fulfill the SHA256 function with ESB.
- Flexibly use WISHBONE bus or HSP bus for the data traffic path.
- Specify the message size in bytes through the parameter.

# 3. Functional Description

## 3.1. Parameter of the Reference Design

**Table 3.1. SHA256 Parameter Description**

| Parameter | Description | Value |
|---|---|---|
| NUM_BYTE | Specify the size of message in bytes. | 1 to $2^{32}$-1 |

## 3.2. Input/Output of the Reference Design

Figure 3.1 is the I/O diagram of the SHA256 reference design.

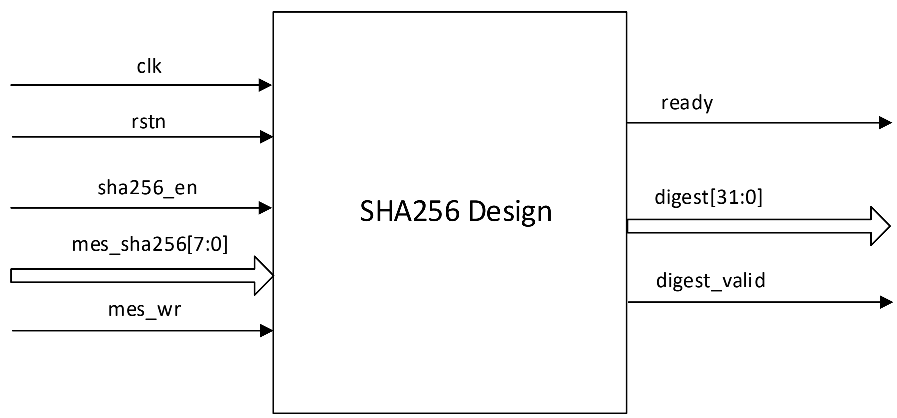


**Figure 3.1. I/O Diagram of SHA256 Reference Design**

Table 3.2 lists the I/O ports of the reference design.

**Table 3.2. Pin Descriptions**

| Signal | Width | Type | Active | Description |
|---|---|---|---|---|
| clk | 1 | Input | Rising edge | System clock |
| rstn | 1 | Input | Low | Asynchronous reset |
| sha256_en | 1 | input | High | Enable signal. When asserted, the reference design runs SHA256 function in ESB. |
| mes_sha256 | 8 | Input | N/A | Data bus with message for SHA256 processing. |
| mes_wr | 1 | Input | High | Message valid signal. When active, message is presented on the data bus. |
| ready | 1 | Output | High | Ready output. When active, it indicates that the design is ready to accept the next message byte. |
| digest | 32 | Output | N/A | Final result with the 32 bytes digest. |
| digest_valid | 1 | Output | High | Digest valid signal. When active, valid digest is on the digest bus. |

## 3.3. Interface with WISHBONE Bus Only

The NUM_BYTE parameter needs to be set based on the size of the message in bytes before running the design. Then the sha256_en signal is asserted to run the design. Detecting the rising edge of sha256_en, the design initializes the ESB for operation, and asserts the ready signal to accept message from the user logic. The user logic needs to provide the message bytes along with the control signal mes_wr. The message is sent to the design using the procedure below:

1. The design informs the user logic that it is ready to receive the new message by asserting the ready signal for one clock cycle.
2. After asserting the ready signal, the design waits for the signal mes_wr to be active. If the design finds the signal mes_wr asserted by the user logic, it takes the signal mes_sha256 as the new message byte.
3. The design counts the size of the message in bytes until it reaches the value of NUM_BYTE. After the last byte of message is delivered, the user logic needs to wait for the signal digest_valid to be asserted.

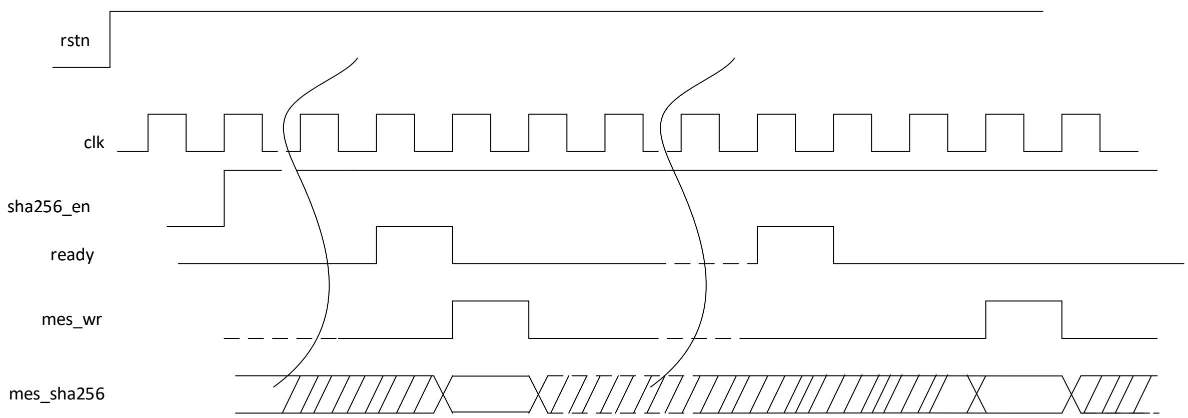


**Figure 3.2. Timing Diagram of Signal ready, Message Valid, and Message for WISHBONE Bus**

The design asserts signal digest_valid to inform the user logic to receive the digest. The signal digest_valid is asserted for eight times to provide the 32 bytes digest, and each time asserted for one clock cycle. The digest is Little Endian.

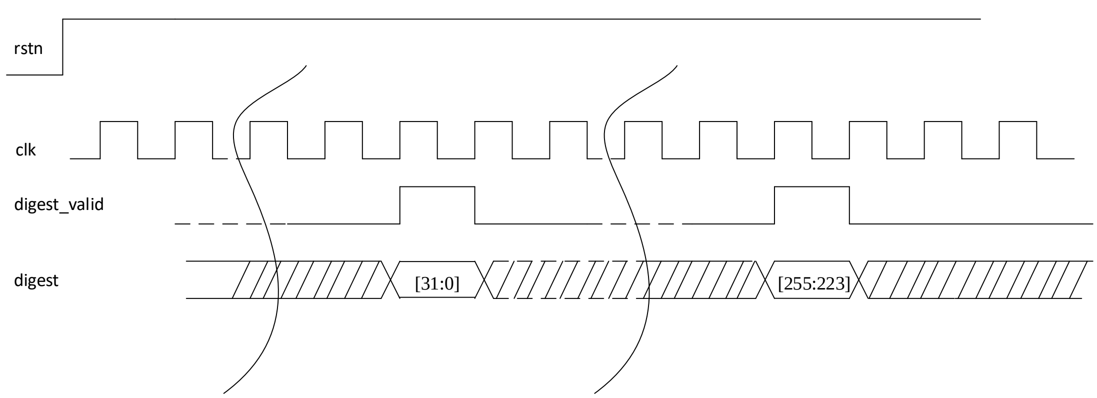


**Figure 3.3. Timing Diagram of Signal digest_valid and digest for WISHBONE Bus**

## 3.4. Interface with WISHBONE Bus and HSP Bus

The NUM_BYTE parameter needs to be set based on the size of the message in bytes before running the design. Then the sha256_en signal is asserted to run the design. Detecting the rising edge of the sha256_en signal, the design initializes the ESB for operation, asserts the ready signal to indicate that it is ready to accept message from the user logic. The user logic needs to provide the message bytes along with the control signal mes_wr. The message is sent to the design using the procedure below:

1. The design informs the user logic that it is ready to receive the new message by asserting the ready signal. The ready signal is consecutive.
2. After asserting the ready signal, the design will wait for the signal mes_wr to be active. If the design finds the signal mes_wr asserted by the user logic, it takes the bus signal mes_sha256 as the new message byte.
3. The design will count the size of the message in bytes until it reaches the value of NUM_BYTE. After the last byte of message is delivered, the user logic needs to wait for the signal digest_valid to be active. The signal digest_valid is consecutive.

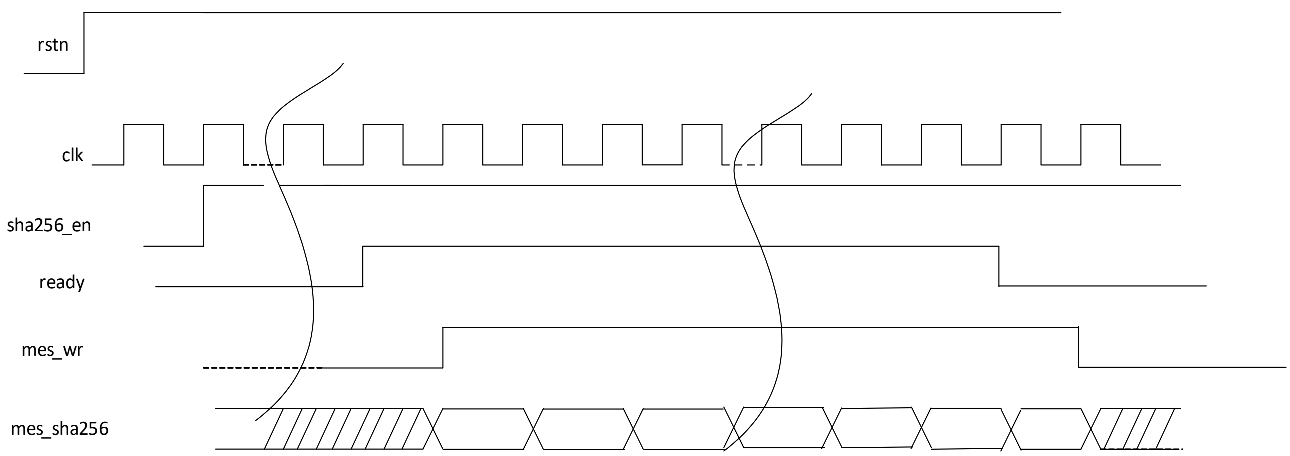


**Figure 3.4. Timing Diagram of Signal ready, Message Valid and Message for HSP Bus**

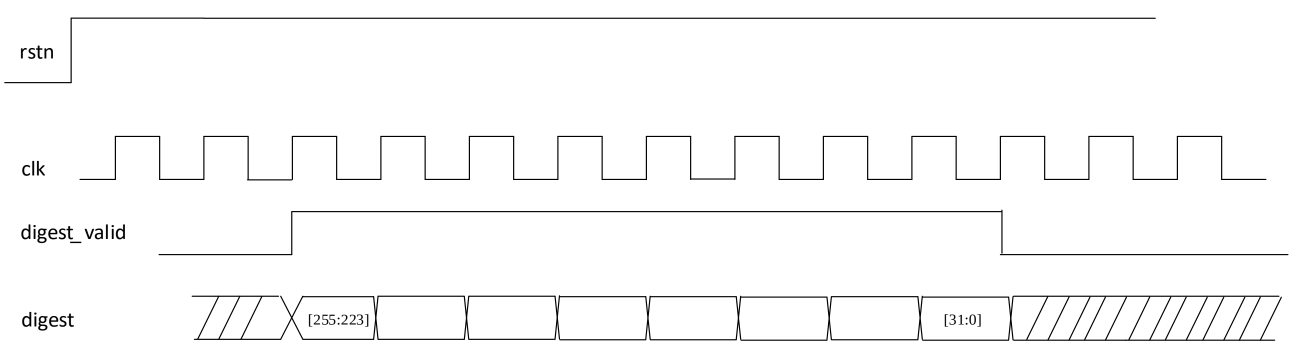


**Figure 3.5. Timing Diagram of Signal Digest_valid and Digest for HSP Bus**

## 3.5. ESB Registers for SHA256

Table 3.1 lists the registers of the reference design.

**Table 3.3. Register Descriptions**

| Register Type | Register Name | Address | Read/Write | Description |
|---|---|---|---|---|
| Control Register | ro_gp0 | 18’h2_0020 | Read | Check for busy status of ESB:<br>0xB0: READY to get a new command<br>0xB2: ESB operation done |
| | as_src_sel | 18’h2_003C | Write | Set SHA256 engine source:<br>[1:0]: 0x2: Use WISBHBONE interface to transmit data<br>0x3: Use HSP interface to transmit data |
| | sha_con | 18'h2_3070 | Write | SHA256 control:<br>[0]: 0x1->0x0: initial SHA256 engine |
| | ri_ctrl1 | 18'h2_000C | Write | ESB function:<br>0x05: Enable SHA256<br>0x00: Disable SHA256 |
| SHA Register | sha_in_byte | 18’h2_304C | Write | Message input data (byte):<br>[31]: last byte indicator<br>[7:0]: payload |
| | sha_res_0 | 18’h2_3050 | Read | Digest result data [31:0] |
| | sha_res_1 | 18’h2_3054 | Read | Digest result data [63:32] |
| | sha_res_2 | 18’h2_3058 | Read | Digest result data [95:64] |
| | sha_res_3 | 18’h2_305C | Read | Digest result data [127:96] |
| | sha_res_4 | 18’h2_3060 | Read | Digest result data [159:128] |
| | sha_res_5 | 18’h2_3064 | Read | Digest result data [191:160] |
| | sha_res_6 | 18’h2_3068 | Read | Digest result data [223:192] |
| | sha_res_7 | 18’h2_306C | Read | Digest result data [255:224] |

The state machine in this reference design runs the ESB using the procedure below (Figure 3.5):

1. Poll the register ro_gp0 in the ESB until the value of this register is 0xB0.
2. Set the register as_src_sel of the ESB to select SHA256 engine data source to be either WISHBONE or HSP.
3. Set the register sha_con to 0x1 and then to 0x0 in order to initialize the ESB’s SHA256 engine.
4. Set register ri_ctrl1 in the ESB to 0x05.
5. Assert the ready signal and wait for the signal mes_wr to be active.
6. If the asserting of the signal mes_wr is detected, receive the message byte until the amount of the message byte is equal to NUM_BYTE.
7. Read register ro_gpo in the ESB until the value of this register is 0xB2.
8. Read the digest from the ESB and assert the signal digest_valid.

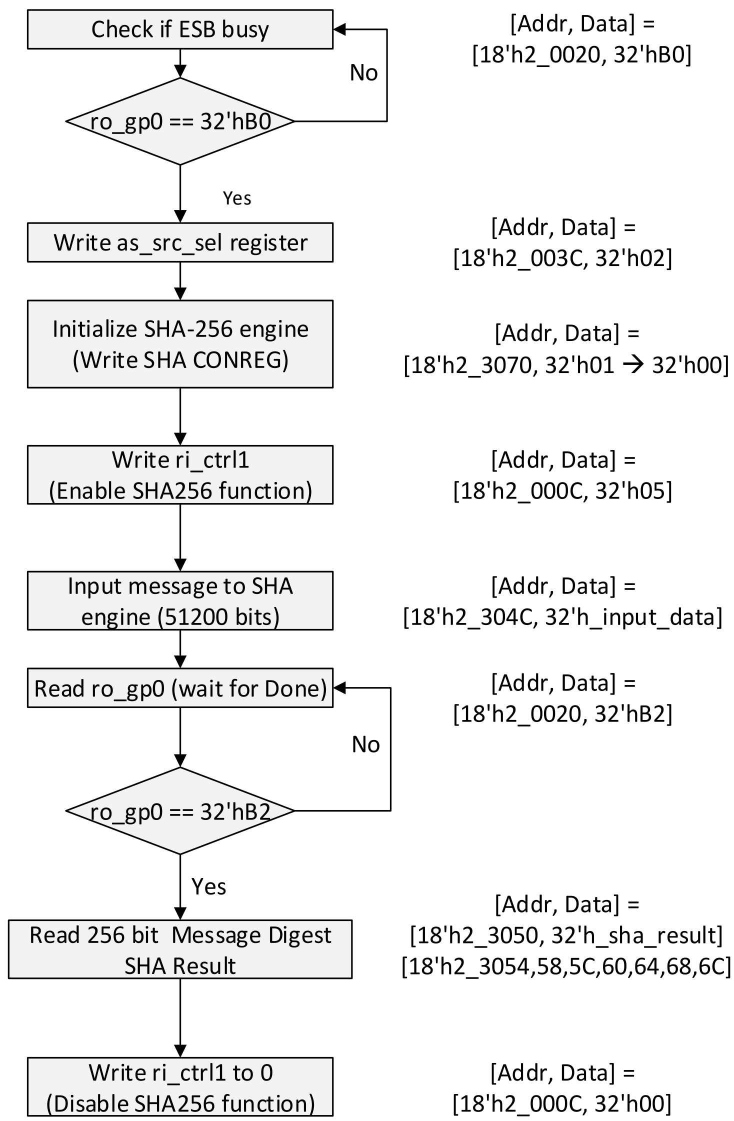


**Figure 3.5. SHA256 Digest Generation Algorithm**

# 4. HDL Simulation and Verification

The simulation takes a golden data as an example. The simulation sets the parameter NUM_BYTE to be 6400. After reset, the test bench asserts sha256_en and then inputs the message (Figure 4.1 and Figure 4.3). The design processes the message and generates the digest output using the ESB engine. The testbench compares the simulation output with the golden output to check the functionality (Figure 4.2 and Figure 4.4).

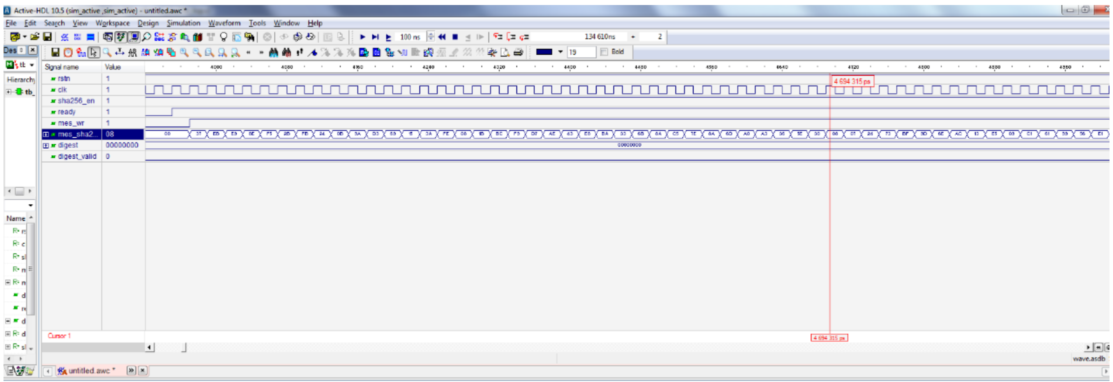

**Figure 4.1.Emulated SHA256 Message Written Transmission with HSP Bus**

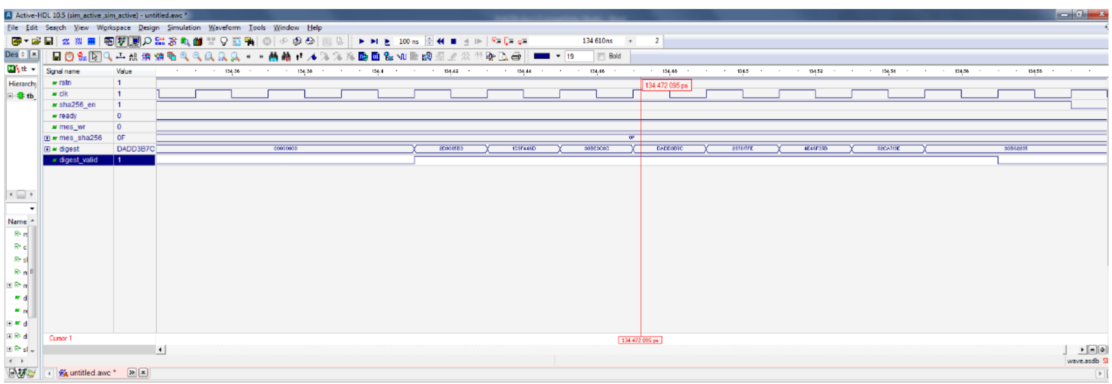

**Figure 4.2. Emulated SHA256 Digest Output Transmission with HSP Bus**

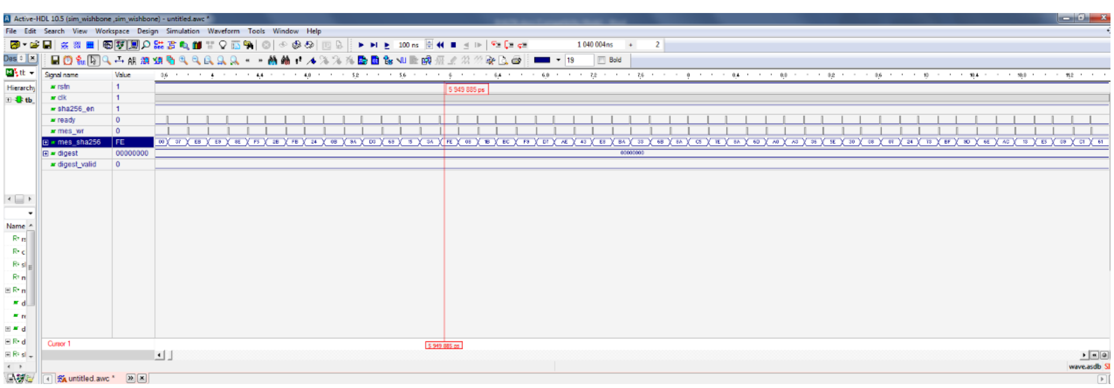

**Figure 4.3. Emulated SHA256 Message Written Transmission with WISHBONE Bus**

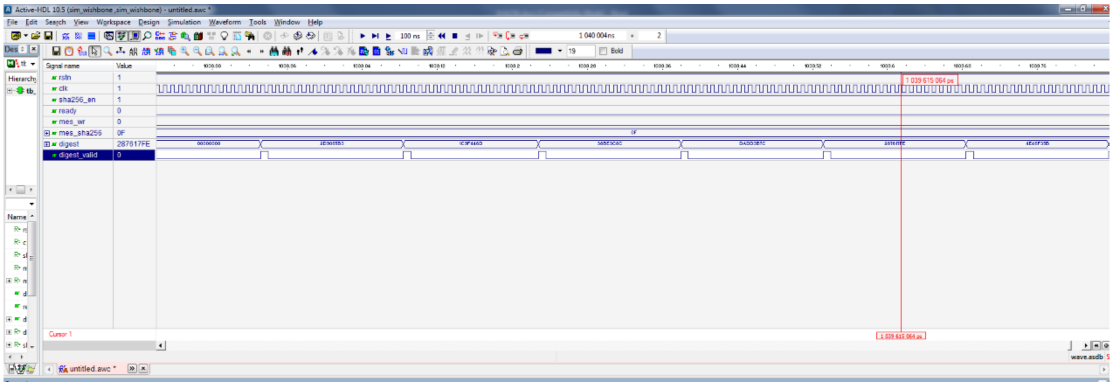


**Figure 4.4. Emulated SHA256 Digest Output Transmission with WISHBONE Bus**

# 5. Implementation

This reference design is implemented in Verilog HDL using Lattice Diamond® software with Synplify Pro® as the synthesis tool. When using this design in a different device, density, or speed, performance and utilization may vary.

**Table 5.1. Performance and Resource Utilization Using WISHBONE Bus Only**

| Device Family | Language | Utilization | Operating Frequency | ESB Primitive | OSC Primitive | Number of I/O |
|---|---|---|---|---|---|---|
| LCMXO3D-9400HC | Verilog HDL | 203 LUTs | >50 MHz | Yes | Yes | 46 |

**Table 5.2. Performance and Resource Utilization Using WISHBONE Bus and HSP Bus**

| Device Family | Language | Utilization | Operating Frequency | ESB Primitive | OSC Primitive | Number of I/O |
|---|---|---|---|---|---|---|
| LCMXO3D-9400HC | Verilog HDL | 195 LUTs | >50 MHz | Yes | Yes | 46 |

**Note**: Performance and utilization characteristics are generated LCMXO3D-9400HC, using Diamond 3.11 design software.

# References

MachXO3D Embedded Security Block (FPGA-TN-02091)

# Technical Support Assistance

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

# Revision History

**Revision 1.0, August 2021**

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

www.latticesemi.com