SPI to WISHBONE Configuration Interface Bridge Usage Guide

Reference Design

FPGA-RD-02191-1.0

March 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
FPGA	Field Programmable Gate Array
IP	Intellectual Property
LUT	Lookup-Table
SPI	Serial Peripheral Interface
WB	WISHBONE Bus
SDA	Serial Data
SCL	Serial Clock
NVM	Non-Volatile Memory (NVCM for MachXO3L, Flash memory for others)
SDM	Self-Download Mode
GPIO	General Purpose Input/Output
EFB	Embedded Function Block

1. Introduction

The SPI to WISHBONE Configuration Interface Bridge Reference Design provides a Slave SPI interface with bridging logic to convert the SPI traffic into WISHBONE bus format. Hence, this allows the external SPI master to access Lattice FPGA configuration logic through the internal fabric WISHBONE interface, which exists on the MachXO2™, MachXO3™, and MachXO3D™ device families. This IP can serve as a replacement for the Slave SPI sysConfig™ interface and provide flexibility to the SPI interface from any available GPIO.

Two main concerns are paramount for the SPI to WISHBONE Configuration Interface Bridge:

- Provide an efficient way to access the configuration NVM in Transparent Mode.
- Flexiable I/O pin assignment, independent from sysConfig pins.

This design is implemented in Verilog. Lattice design tools are language agnostic, so the Verilog design can be seamlessly added to a VHDL design if needed. The Lattice Diamond® Software Place and Route tool integrated with Synplify Pro® synthesis tool is used for design implementation. The design can be targeted to all MachXO2, MachXO3, and MachXO3D family devices.

1.1. Quick Facts

Table 1.1 presents a summary of the SPI to WISHBONE Configuration Interface Bridge Reference Design.

Table 1.1. Quick Facts

IP Requirements	Supported FPGA Families	MachXO2, MachXO3, MachXO3D
	Minimal Device Needed	MachXO2-256
Device Requirements	Targeted Device	Any device in the MachXO2, MachXO3, and MachXO3D families
	Supported User Interfaces	WISHBONE Bus Interface
Design Tool Support	Lattice Implementation	Lattice Diamond Software 3.11 or higher
	Synthesis	Lattice Synthesis Engine (LSE)
		Synopsys® Synplify Pro for Lattice
	Simulation	For the list of supported simulators, see the Lattice Diamond Software 11.0 User Guide.

1.2. Conventions

1.2.1. Definition of Terms

This document uses the following terms to describe common functions:

- Configuration – Refers to a change in the state of the MachXO3D SRAM memory cells.
- Transparent mode – Used to update the Configuration Flash and the User Flash Memory while leaving the MachXO3D device in user mode.
- User mode – The MachXO3D device is in user mode when configuration is complete and the FPGA is performing the logic functions it is programmed to perform.

1.2.2. Nomenclature

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

1.2.3. Signal Names

Signal names that end with:

- _*n* are active low (asserted when value is logic 0)
- _*i* are input signals
- _*o* are output signals

2. Features

The SPI to WISHBONE Configuration Interface Bridge behaves as a slave on the SPI bus. It passes the SPI data packet onto the built-in WISHBONE bus of the MachXO2, MachXO3, or MachXO3D device, which is a system configuration interface. Therefore, you can establish NVM programming in the user functional mode.

The SPI to WISHBONE Configuration Interface Bridge provides the following features and requirements:

- Serial SPI interface up to 22 MHz
- The WISHBONE clock must be at least 3X of the SPI bus clock
- The SPI CSn de-assertion interval must be at least 1 SPI clock period duration
- SPI mode 0 is required (Launch data on SCLK Falling Edge, Capture data on SCLK Rising Edge)

3. Signal Description

Table 3.1 lists and describes the input and output signals of the SPI to WISHBONE Configuration Interface Bridge.

Table 3.1. Signal Description

Signal	Direction	Description
SPI Interface		
SCLK	Input	Slave SPI clock
SI	Input	Slave SPI Serial Data Input
SO	Output	Slave SPI Serial Data output
SSn	Input	Slave SPI Chip Select, Active low
WISHBONE Interface		
wb_rst_i	Input	WISHBONE Reset Resets the WISHBONE interface and sets registers to their default values. Also, reset the internals of the IP block.
wb_clk_i	Input	WISHBONE Clock Clock for the WISHBONE interface and the rest of the IP
wb_we_o	Output	WISHBONE Write Enable
wb_adr_o[7:0]	Output	WISHBONE E Address
wb_dat_o[7:0]	Output	WISHBONE Output Data
wb_dat_i[7:0]	Input	WISHBONE Input Data
wb_str_o	Output	WISHBONE Strobe
wb_cyc_o	Output	WISHBONE Cycle (= Strobe)

4. Functional Overview

The SPI to WISHBONE Configuration Interface Bridge supports standard SPI communication transactions, on the four-wire interface, SCLK clock input, SI slave data input, SO slave data output, and SSn slave chip select input. The data packets between the SPI slave interface inside the Reference Design and the external SPI master go through the bridging logic and feed into the WISHBONE Configuration Interface to invoke the device configuration logic to access the on chip Non-Volatile Memory. The bridging logic handles the WISHBONE addressing for the WISHBONE Configuration Interface.

The SPI to WISHBONE Configuration Interface Bridge functional diagram is shown in Figure 4.1.

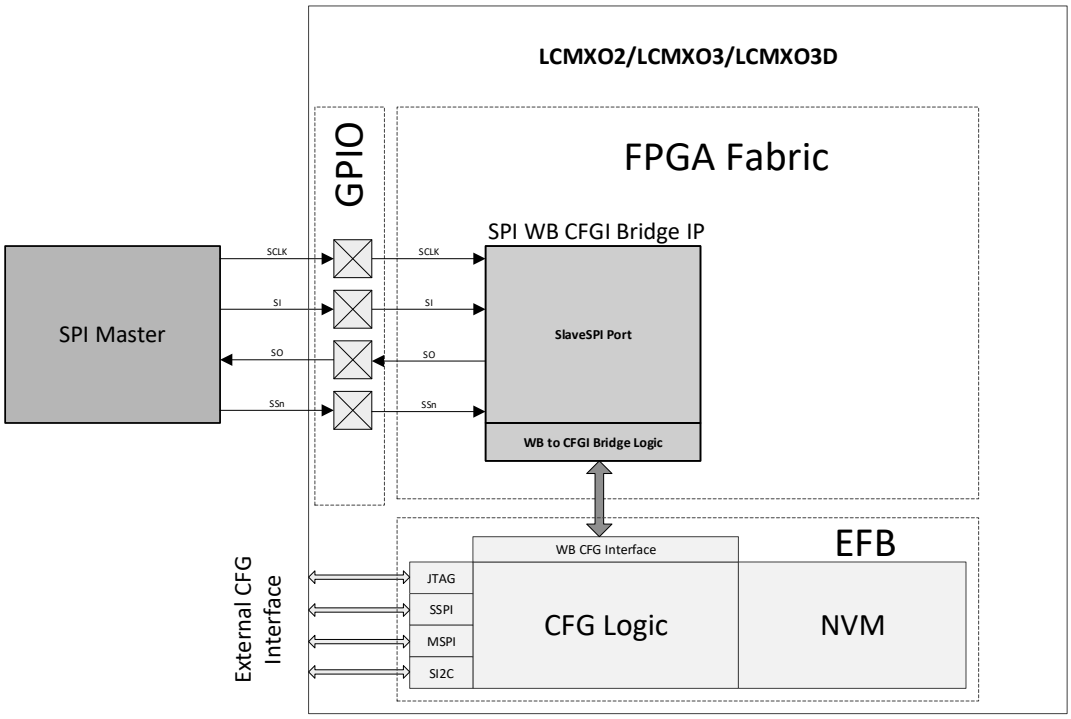

Figure 4.1. SPI to WISHBONE Configuration Interface Bridge Functional Diagram

The SPI to WISHBONE Configuration Interface Bridge pin diagram is shown in Figure 4.2 below.

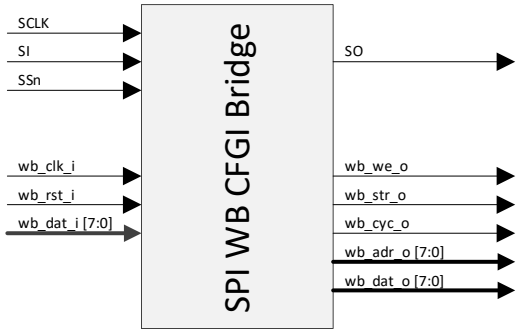

Figure 4.2. . SPI to WISHBONE Configuration Interface Bridge Pin Diagram

5. Design Instantiation, Simulation, and Validation

This section provides information on how to instantiate the SPI to WISHBONE Configuration Interface Bridge Reference Design, including using the Lattice Diamond Software to generate the EFB, and how to run simulation and synthesis. For more details on the Lattice Diamond Software, refer to the Lattice Diamond Software 3.11 User Guide.

5.1. EFB Generation

The SPI to WISHBONE Configuration Interface Bridge Reference Design can access the Configuration Flash, User Flash Memory, and the Feature Row from an internal WISHBONE bus. To use the WISHBONE bus, the Embedded Function Block must be inserted into your design. This allows for flexible utilization of the EFB, for example, if your design makes use of other features of the EFB such as UFM. If no other features are required, the EFB can be generated as shown below. In either case, insert the generated EFB instantiation into your design and hook up the WISHBONE bus to the Bridge reference design.

To configure the EFB hard IP functions and generate the EFB module using IPExpress:

1. From the Lattice Diamond® top menu select **Tools > IPExpress**.
2. With a device targeted for the Diamond project, the IPExpress window opens and the EFB module is found under **Modules > Architecture Modules**.

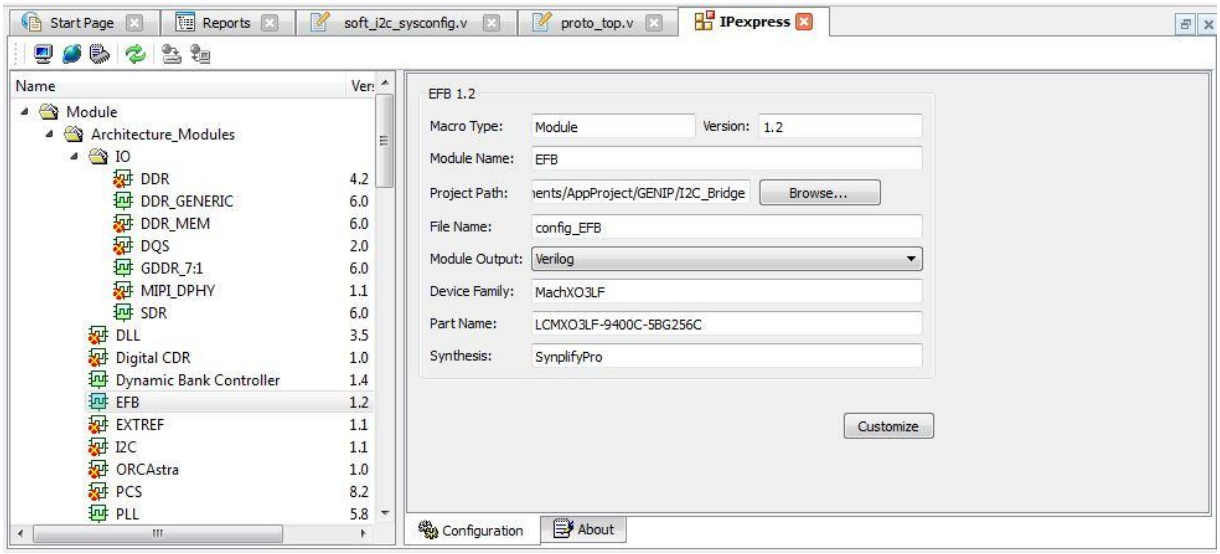

Figure 5.1. EFB Module in IPExpress

3. Fill in the **Project Path**, **File Name**, and **Design Entry** fields, and click **Customize**.
4. The EFB configuration dialog appears. The left side of the EFB window displays a graphical representation of the I/O associated with each IP function. The I/O pins appear and disappear as each IP is enabled or disabled. Check the tab for **WISHBONE**, and enter the **WISHBONE Clock Frequency**. An example EFB with all features enabled is shown in Figure 5.2.
5. Click the **Generate** button.

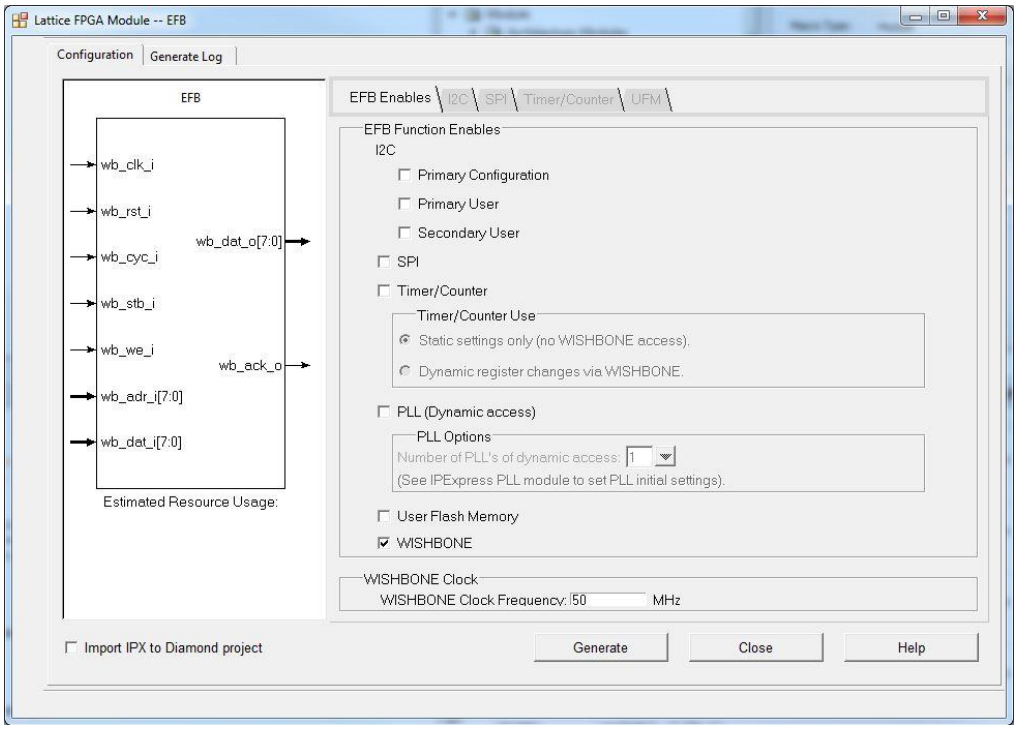

Figure 5.2. Generating an EFB Module wit IPExpress

The EFB module is generated and ready to be instantiated in your design.

5.2. IP Module Declaration and Instantiation

The SPI to WISHBONE Configuration Interface Bridge can be instantiated into the RTL design, as shown in the example below.

IP Module Declaration

```
module SSPI2WB_birdge (SCLK, SI, SO, SSn, wb_clk_i, wb_rst_i, wb_we_o, wb_adr_o,
                                                                    wb_dat_o,
wb_dat_i, wb_str_o, wb_cyc_o);
input SCLK;
input SI;
output SO;
input SSn;
input wb_clk_i;
input wb_rst_i;
output wb_we_o;
output [7:0] wb_adr_o;
output [7:0] wb_dat_o;
input [7:0] wb dat i;
output wb_str_o;
output wb_cyc_o;
endmodule
```

```
IP Instantiation
wire                wb_clk_i  ;
wire                wb_rst_i  ;
wire                wb_we_i   ;
wire    [7:0]    wb_adr_i ;
wire    [7:0]    wb_dat_i ;
wire    [7:0]    wb_dat_o ;
wire                wb_str_i ;
wire                wb_cyc_i ;
wire                wb_ack_o ;

// SPI to WISHBONE Configuration Bridge
SSPI2WB_birdge SSPIWB_bridge_inst (
        .SCLK    (SCLK   ),
        .SI    (SI   ),
        .SO  (SO ),
        .SSn (SSn),
        .wb_clk_i  (wb_clk_i),
        .wb_rst_i  (wb_rst_i),
        .wb_we_o   (wb_we_i ),
        .wb_adr_o  (wb_adr_i),
        .wb_dat_o  (wb_dat_i),
        .wb_dat_i  (wb_dat_o),
        .wb_str_o  (wb_str_i),
        .wb_cyc_o  (wb_cyc_i));

// MachXO EFB
config_EFB EFB_inst (
       .wb_clk_i(wb_clk_i ),
       .wb_rst_i(wb_rst_i ),
       .wb_we_i  (wb_we_i  ),
       .wb_adr_i(wb_adr_i ),
       .wb_dat_i(wb_dat_i ),
       .wb_dat_o(wb_dat_o ),
       .wb_stb_i(wb_str_i ),
       .wb_cyc_i(wb_cyc_i ),
       .wb_ack_o(wb_ack_o ));
```

5.3. WISHBONE Configuration Commands

The WISHBONE Configuration commands consist of one byte mandatory OPCODE, three bytes mandatory OPERANDs (except for ISC_NOOP command which is OPCODE only), and optional data bytes and dummy bytes which are defined on a per command basis as shown in Table 5.1 below.

Table 5.1. WISHBONE Configuration Command Format

OPCODE (1 Byte)	OPERAND1 (1 Byte)	OPERAND2 (1 Byte)	OPERAND3 (1 Byte)	DATA (X Bytes)
Mandatory	Mandatory	Mandatory	Mandatory	Optional

The supported WISHBONE Configuration Commands for the MachXO2, MachXO3, and MachXO3D product families are shown in Table 5.2.

Table 5.2. WISHBONE Configuration Command Table

Command Name [SVF Synonym]	Command	Operands	Write Data	Read Data	Notes
Read Device ID [IDCODE_PUB]	0xE0	00 00 00	N/A	YY YY YY YY	YY characters represent the device-specific ID code
Enable Configuration Interface (Transparent Mode) [ISC_ENABLE_X]	0x74	08 00 00 [1]	N/A	N/A	Enable the Configuration Logic for device programming in transparent mode. [1]
Read Busy Flag [LSC_CHECK_BUSY]	0xF0	00 00 00	N/A	YY	Bit 1 0 7 Busy Ready
Read Status Register [LSC_READ_STATUS]	0x3C	00 00 00	N/A	YY YY YY YY	Bit 1 0 12 Busy Ready 13 Fail OK
Erase [ISC_ERASE]	0x0E	0Y 00 00	N/A	N/A	Y = Memory space to erase Y is a bitwise OR Bit 1=Enable Bit 1=Enable 16 Erase SRAM 17 Erase Feature Row 18 Erase Configuration Flash 19 Erase UFM
Erase UFM [LSC_ERASE_TAG]	0xCB	00 00 00	N/A	N/A	Erase the UFM sector only.
Reset Configuration Flash Address [LSC_INIT_ADDRESS]	0x46	00 00 00	N/A	N/A	Set Page Address pointer to the beginning of the Configuration Flash sector
Set Address [LSC_WRITE_ADDRESS]	0xB4	00 00 00	M0 00 PP PP	N/A	Set the Page Address pointer to the Flash page specified by the least significant 14 bits of the PP PP field. The 'M' field defines the Flash memory space to access. Field 0x0 0x4 M Configuration Flash UFM
Program Page [LSC_PROG_INCR_NV]	0x70	00 00 01	YY * 16	N/A	Program one Flash page. Can be used to program the Configuration Flash, or UFM.
Reset UFM Address [LSC_INIT_ADDR_UFM]	0x47	00 00 00	N/A	N/A	Set the Page Address Pointer to the beginning of the UFM sector
Program UFM Page [LSC_PROG_TAG]	0xC9	00 00 01	YY * 16	N/A	Program one UFM page
Program USERCODE [ISC_PROGRAM_USERCODE]	0xC2	00 00 00	YY * 4	N/A	Program the USERCODE.
Read USERCODE [USERCODE]	0xC0	00 00 00	N/A	YY * 4	Retrieves the 32-bit USERCODE value
Write Feature Row [LSC_PROG_FEATURE]	0xE4	00 00 00	YY * 8	N/A	Program the Feature Row bits
Read Feature Row [LSC_READ_FEATURE]	0xE7	00 00 00	N/A	YY * 8	Retrieves the Feature Row bits
Write FEABITS	0xF8	00 00 00	YY * 2	N/A	Program the FEABITS

Command Name [SVF Synonym]	Command	Operands	Write Data	Read Data	Notes
[LSC_PROG_FEABITS]					
Read FEABITS [LSC_READ_FEABITS]	0xFB	00 00 00	N/A	YY * 2	Retrieves the FEABITS
Read Flash [LSC_READ_INCR_NV]	0x73	M0 PP PP	N/A	YY * N[2] * PPPP	Retrieves PPPP count pages. Only the least significant 14 bits of PP PP are used. The 'M' field must be set based on the con-figuration port being used to read the Flash memory. 0x0 I2C 0x1 JTAG/SSPI/WB
Program DONE [ISC_PROGRAM_DONE]	0x5E	00 00 00	N/A	N/A	Program the DONE status bit enabling SDM
Program OTP Fuses [LSC_PROG_OTP]	0xF9	00 00 00	UCFSUCFS	N/A	Makes the selected memory space One Time Programmable. Matching bits must be set in unison to activate the OTP feature. Bit 1 0 0, 4 SRAM OTP SRAM Writable 1, 5 Feature OTP Feature Writable 2, 6 CF OTP CF Writable 3, 7 UFM OTP UFM Writable
Disable Configuration Interface [ISC_DISABLE]	0x26	00 00	N/A	N/A	Exit Transparent programming mode.
Bypass [ISC_NOOP]	0xFF	N/A	N/A	N/A	No Operation
Refresh [LSC_REFRESH]	0x79	00 00	N/A	N/A	Force the MachXO2 to reconfigure. Trans-mitting a REFRESH command reconfigures the MachXO2 in the same fashion as asserting PROGRAMN.
Program SECURITY [ISC_PROGRAM_SECURITY]	0xCE	00 00 00	N/A	N/A	Program the Security bit (Secures CFG Flash sector).[3]
Program SECURITY PLUS [ISC_PROGRAM_SECPLUS]	0xCF	00 00 00	N/A	N/A	Program the Security Plus bit (Secures CFG and UFM Sectors).[3]
Read TraceID code [UIDCODE_PUB]	0x19	00 00 00	N/A	YY * 8	Read 64-bit TraceID.

Notes:

1. Transmit the command opcode and first two operand bytes when using the I[2]C port. The final operand byte must not be transmitted.
2. The N is determined by the DSR size of the device, which N = Total DSR Bits/8.
3. SECURITY and SECURITY PLUS commands are mutually exclusive.

For details on the Non-Volatile Memory Programming Flow, refer to:

- MachXO2 Programming and Configuration Usage Guide (FPGA-TN-02155)
- MachXO3 Programming and Configuration Usage Guide (FPGA-TN-02055)
- MachXO3D Programming and Configuration Usage Guide (FPGA-TN-02069)

5.4. SPI Transmission Format]

The typical WISHBONE Configuration write operation through the SPI to WISHBONE Configuration Interface Bridge is shown in Figure 5.3.

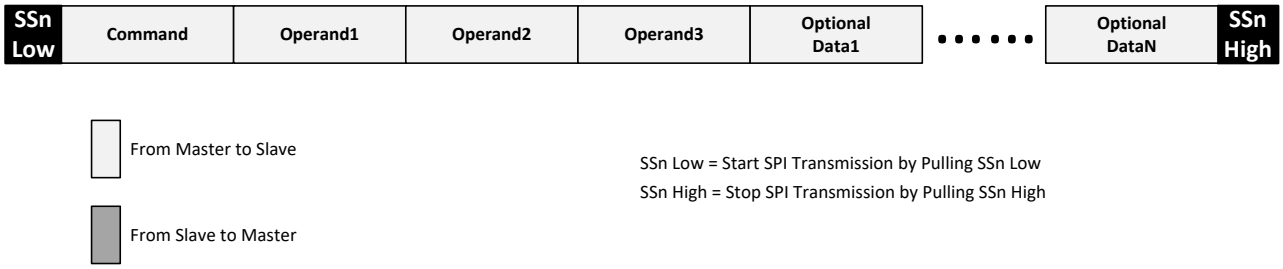

Figure 5.3. SPI Transmission Format for Write

The typical WISHBONE Configuration read operation through the SPI to WISHBONE Configuration Interface Bridge is shown in Figure 5.4.

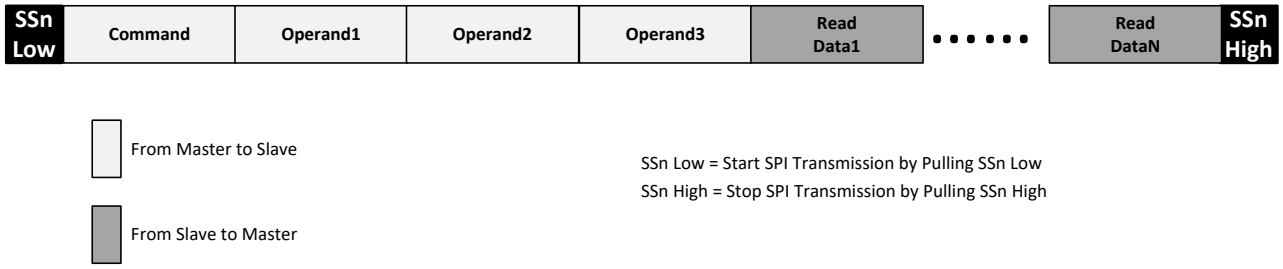

Figure 5.4. SPI Transmission Format for Read

5.5. Design Simulation

The simulation environment and test bench *tb_spi_bridge.v* instantiates the *SPI2WB_bridge_sysconfig.v*, which contains the SPI to WISHBONE Configuration Interface Bridge module and the FPGA device EFB module.

To simulate the design:

1. Invoke the entire simulation setup by opening the project file *Project/SPI_Bridge_sim.ldf* from Lattice Diamond. The SPI to WISHBONE Configuration Interface Bridge inputs are driven with a generated pattern from the SPI test bench.

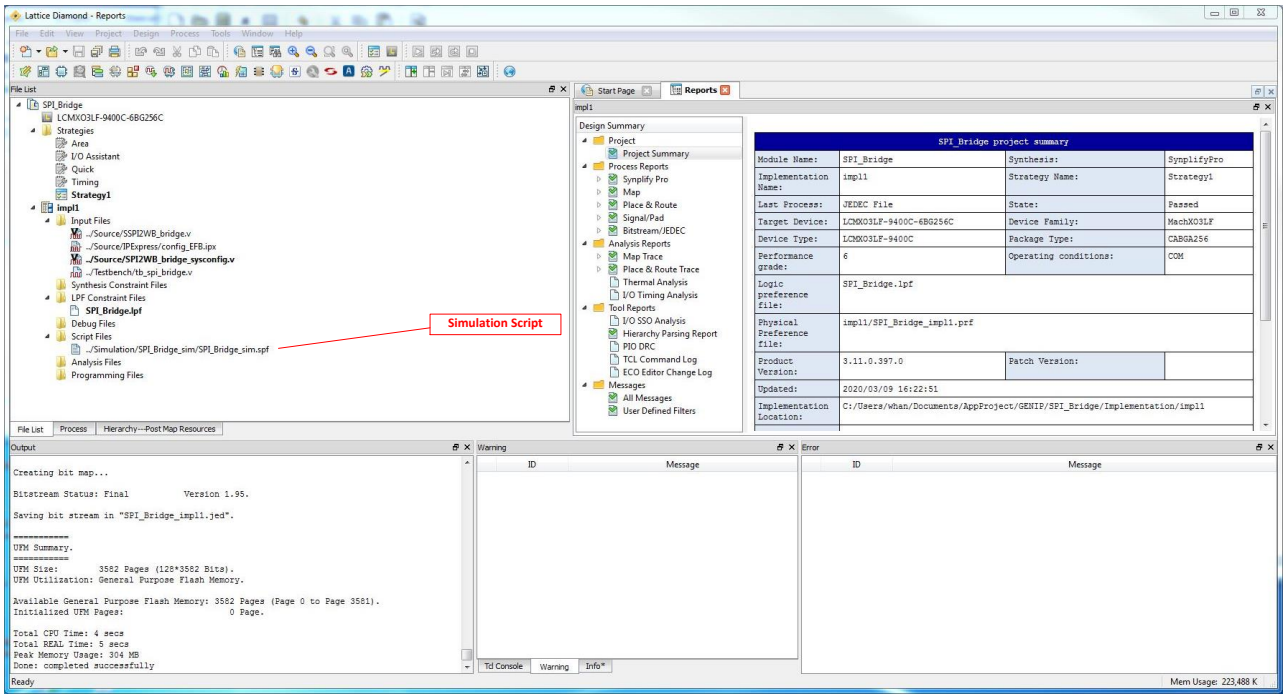

Figure 5.5. Simulation Project in Lattice Diamond

2. Access the simulation environment by double clicking on the *SPI_Bridge_sim.spf* script file in Lattice Diamond from the file list.
3. Click **OK**. The Aldec ActiveHDL opens.
4. Compile the project and initialize the simulation.
5. Add signals to the waveform viewer to view and run the simulation.

The testbench shows the following statements while running:

```
run @30us
# KERNEL: Configuration complete. EFB now in user mode.          820
# KERNEL:
# KERNEL:          1000  tb_spi_bridge          << Test Read Device ID >>
# KERNEL:          1000                       << Command + Operands = E0-00-00-00 >>
# KERNEL:          3838                       << READ VERIFIED: Received Data = 012b2043 >>
# KERNEL:
# KERNEL:          3882  tb_spi_bridge          << Assert Enable X >>
# KERNEL:          3882                       << Command + Operands = 74-00-00-00 >>
# KERNEL:
```

```
# KERNEL:          5356  tb_spi_bridge          << Test Read Status Reg >>
# KERNEL:          5356                        << Command + Operands = 3C-00-00-00 >>
# KERNEL:          8194                        << READ VERIFIED: Received Data = 00201e09 >>
# KERNEL:
# KERNEL:          8238  tb_spi_bridge          << Assert Init Addr >>
# KERNEL:          8238                        << Command + Operands = 46-00-00-00 >>
# KERNEL:
# KERNEL:          9712  tb_spi_bridge          << Test Check BUSY >>
# KERNEL:          9712                        << Command + Operands = F0-00-00-00 >>
# KERNEL:         11494                        << READ VERIFIED: Received Data = 00 >>
# KERNEL:
# KERNEL:         11538  tb_spi_bridge          << Assert Disable >>
# KERNEL:         11538                        << Command + Operands = 26-00-00-00 >>
# KERNEL:
# KERNEL:         13012  tb_spi_bridge          << Assert Bypass >>
# KERNEL:         13012                        << Command + Operands = FF-00-00-00 >>
# KERNEL:
# KERNEL:         14486  tb_spi_bridge          << Simulation complete... >>
# KERNEL:         14486  tb_spi_bridge          << Number of errors:         0 >>
# RUNTIME: Info: RUNTIME_0070 tb_spi_bridge.v (160): $stop called.
# KERNEL: Time: 14486 ns,  Iteration: 0,  Instance: /tb_spi_bridge,  Process: @INITIAL#114_2@.
# KERNEL: Stopped at time 14486 ns + 0.
```

An example of the simulation waveform for transmitting the transparent flash access enable command is shown in Figure 5.6. In addition, an example of the simulation for reading the device ID is shown in Figure 5.7 below.

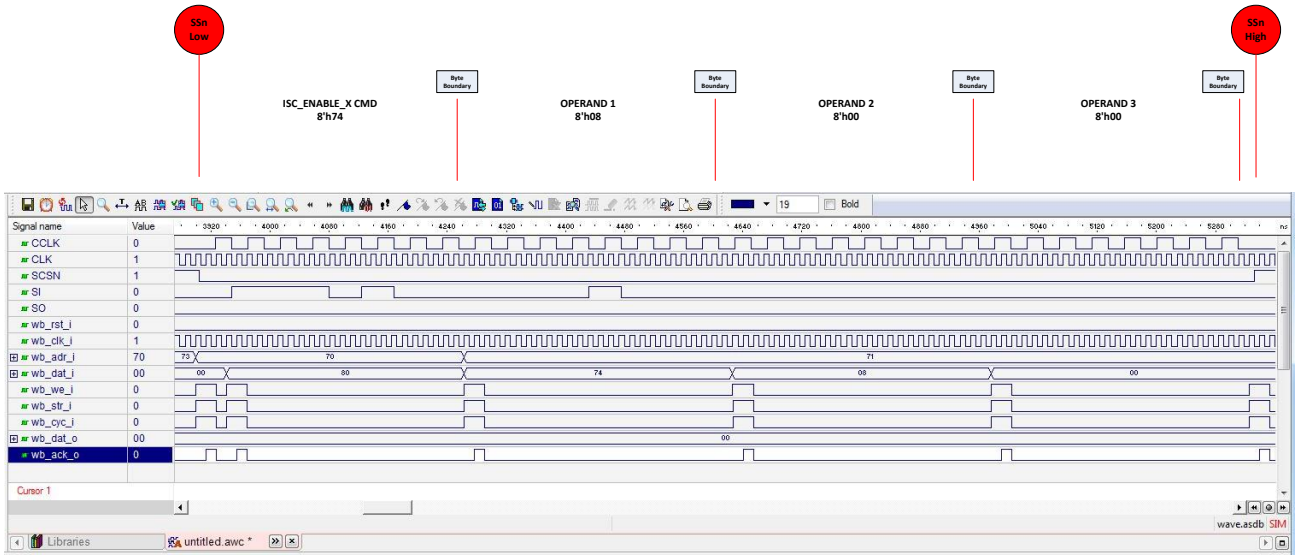

Figure 5.6. Simulation Waveform for Transparent Flash Access Enable

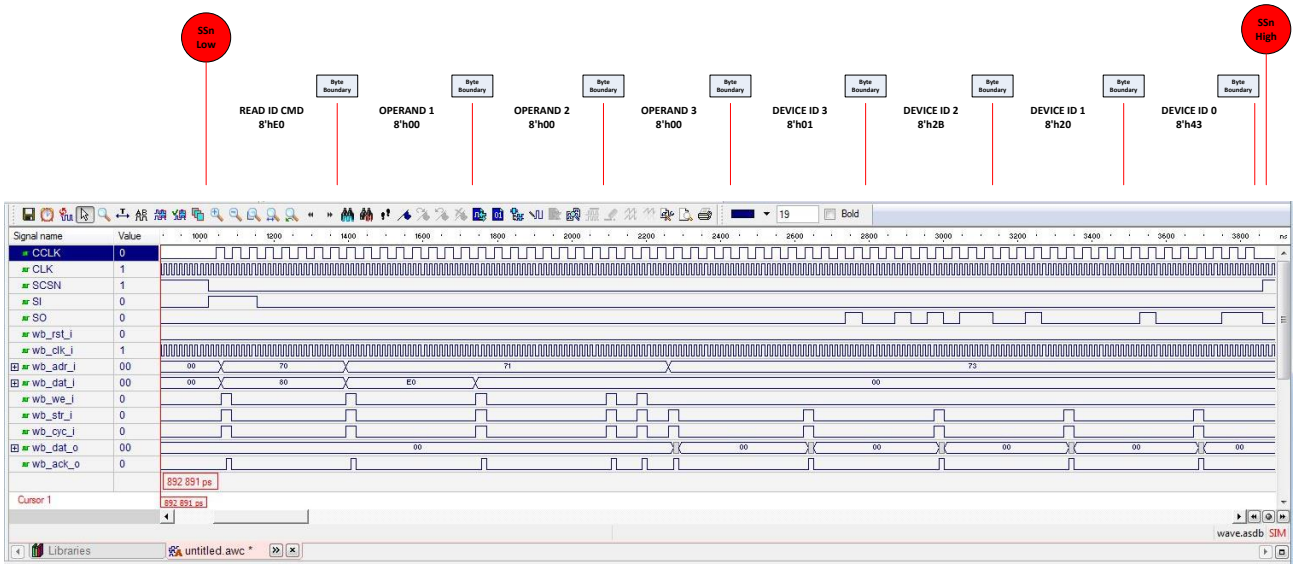

Figure 5.7. Simulation Waveform for Transparent Flash Access Enable

6. Implementation

Resource utilization information for a sample Slave configuration using Lattice Diamond Software is presented below. For more information on Lattice Diamond Software, visit the Lattice web site at www.latticesemi.com/Products/DesignSoftwareAndIP.

Table 6.1. Performance and Resource Utilization

Device Family	Language	Speed Grade	Utilization	Fmax (MHz)	I/O	Architecture Resources
MachXO2	Verilog-LSE	-6	62 LUTs	80	4	EFB
	Verilog-Syn	-6	65 LUTs	80	4	EFB
MachXO3LF	Verilog-LSE	-6	62 LUTs	80	4	EFB
	Verilog-Syn	-6	65 LUTs	80	4	EFB
MachXO3D	Verilog-LSE	-6	62 LUTs	80	4	EFB
	Verilog-Syn	-6	65 LUTs	80	4	EFB

References

For complete information on Lattice Diamond Project-Based Environment, Design Flow, Implementation Flow and Tasks, as well as on the Simulation Flow, see the Lattice Diamond Software 3.11 User Guide.

The SPI bus timing and timing diagrams are described in the *Serial Peripheral Interface (SPIV3) Block Description* document from *Freescale Semiconductor*.

Technical Support Assistance

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

Revision History

Revision 1.0, March 2020

Section	Change Summary
All	Initial release

www.latticesemi.com