M-PESTI Initiator Reference Design

Reference Design

FPGA-RD-02341-1.0

April 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.

Abbreviations	Definition
APB	Advanced Peripheral Bus
CPLD	Complex Programmable Logic Device
CRC	Cyclic Redundancy Check
DC-MHS	Datacenter Modular Hardware System
EBR	Embedded Block RAM
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
JTAG	Joint Test Action Group
LED	Light-Emitting Diode
LUT	Look-Up Table
OCP	Open Compute Project
PLL	Phase-Locked Loop
RTL	Register Transfer Level
SoC	System on Chip
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
VW	Virtual Wire

1. Introduction

M-PESTI is a generic and extensible 1-wire, bidirectional circuit and protocol for applications such as cabled high-speed I/O interposers, managed power distribution, cooling subsystems, and control panels. This reference design provides M-PESTI as an SoC-based project and this reference design is OCP Ready™.

This version of the M-PESTI Initiator reference design provides a Lattice Propel™ Builder software solution template that uses the M-PESTI Initiator IP core with an external I2C driver reference code, and M-PESTI target test component. The reference design is compliant with the M-PESTI Base Specification, which is part of the DC-MHS version 1.0 specification. This reference design includes the following collaterals:

- The SoC design project on which you can open, view, and modify the design through the Lattice Propel Builder software. The bitstream can be generated using the Lattice Diamond™ software.
- The CPLD demonstration bitstream.

1.1. Quick Facts

Download the reference design files from the Lattice reference design web page.

Table 1.1. Summary of the Reference Design

General	Target devices	MachXO3LF™, MachXO5™-NX, MachXO3D™
	Source code format	Verilog, System Verilog, Python
Simulation	Functional simulation	Performed
	Timing simulation	Not performed
	Testbench	Available
	Testbench format	System Verilog
Software Requirements	Software tool and version	• Lattice Radiant™ software 2025.2 or higher • Lattice Diamond™ 3.14 or higher • Lattice Diamond™ Programmer version 3.14 or higher • Lattice Propel™ 2025.2 or higher
	IP version (if applicable)	IP core v2.0.1

Hardware Requirements	Board	• M-PESTI Initiator:
		  • MachXO3LF Starter Kit (LCMXO3LF-6900C-S-EVN) or MachXO3D Breakout Board (LCMXO3D-9400HC-B-EVN) or
		  • MachXO5-NX Development Board (LFMXO5-25-EVN)
		• M-PESTI Target:
		  • MachXO3D Breakout Board (LCMXO3D-9400HC-B-EVN)
		• Aardvark I2C/SPI Host Adapter

	Cable	• Two USB Type-A plug to USB Mini-B plug. • Five port-to-port jumper wires.

1.2. Features

Key features of the M-PESTI Initiator reference design include the following:

- Communicates via a half-duplex, bidirectional UART protocol at 250k-baud rate, 8-bit data, 1-bit odd parity, 1 start bit, and 1 stop bit on the M-PESTI port
- Supports static discovery payload with CRC-8 payload checksum
- Supports a one initiator to many targets system
- Supports a configurable number of M-PESTI devices, up to 64 targets
- Supports an autonomous static discovery payload request command issued to all targets in a round-robin manner during the Discovery Phase
- Supports static discovery payload request command retries. If the payload is not successfully received after an initial attempt, up to two more retries per target are triggered before proceeding to the next target(s). When the execution returns to the target, the initiator continues sending commands as a set of initial attempt followed by up to two retries until payload is successfully received
- Supports target reset at any time
- Supports sending broadcast commands
- Supports aborting an ongoing Discovery or Active Phase command to insert a broadcast command
- Supports source and destination cable coupling discovery

1.3. Limitations

- The M-PESTI Initiator IP core supports MachXO5-NX, MachXO3D, MachXO3L™, MachXO3LF, Certus™-N2, and Avant™ devices. However, this reference design only includes an M-PESTI Initiator project for MachXO3LF, MachXO3D, and MachXO5-NX devices.
- The M-PESTI Initiator IP core supports both Specification 1.0 and 1.2. In this reference design, the I2C sequences and the M-PESTI pseudo-target still follow the Discovery Payload format based on Specification 1.0. This is because Specification 1.2 does not yet fully define the number of Virtual Wires supported by an M-PESTI target based on its Discovery Payload. In Specification 1.0, the number of Virtual Wires supported by an M-PESTI Target is described in Byte 03h of its Discovery Payload.

2. Directory Structure and Files

The following figure shows the directory structure of the reference design.

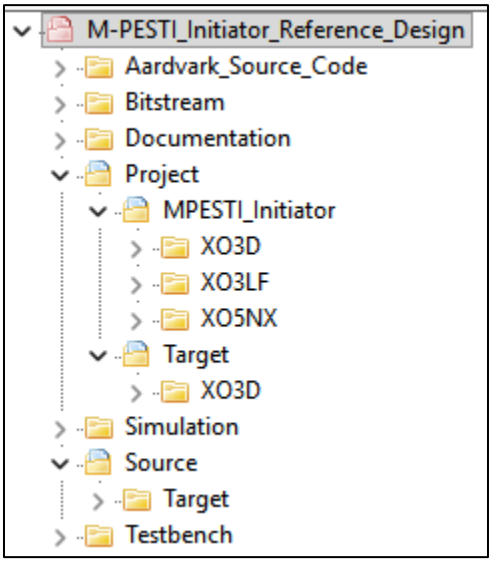

Figure 2.1. Directory Structure

The M-PESTI Initiator reference design package includes the following subfolders:

- **Aardvark_Source_Code** — Contains the Python source codes for both the Command Terminal and GUI-based access. These source codes utilize Aardvark API.
- **Bitstream** — Contains the programming files for the MachXO3LF Starter Kit (Initiator), MachXO5-NX Development Board (Initiator), MachXO3D Breakout Board (Initiator), and MachXO3D Breakout Board (Target)
- **Project** — Contains the Diamond and SoC projects for the M-PESTI Initiator. This folder also contains the Diamond project for the M-PESTI pseudo-target model.
- **Simulation** — Contains the simulation file (*.do*) used to run RTL simulation in the ModelSim™ Lattice FPGA Edition software
- **Source** — Contains the source files for the M-PESTI pseudo-target model
- **Testbench** — Contains the testbench files used in simulation

The following table shows the list of files included in the reference design.

Table 2.1. File List

Filename	Description
mpesti_initiator0.ipx	Contains the information on the files associated to the generated IP.
mpesti_initiator0.cfg	Contains the parameter values used in IP configuration.
mpesti_initiator0.sv	Provides an example RTL top file that instantiates the module.
mpesti_initiator0_bb.v	Provides the synthesis closed-box.
mpesti_initiator0_tmpl.v *mpesti_initiator0_tmpl.vhd*	These files provide instance templates for the module.

3. Functional Description

The following figure shows the high-level block diagram of the reference design, which consists of three key hardware components:

- An M-PESTI Initiator SoC design that is programmed into the MachXO3LF Starter Kit, MachXO3D Breakout Board, or MachXO5-NX Development Board
- Emulated multiple target devices that are programmed into a MachXO3D Breakout Board
- A PC-based Command Terminal/GUI-based access using the reference Python code

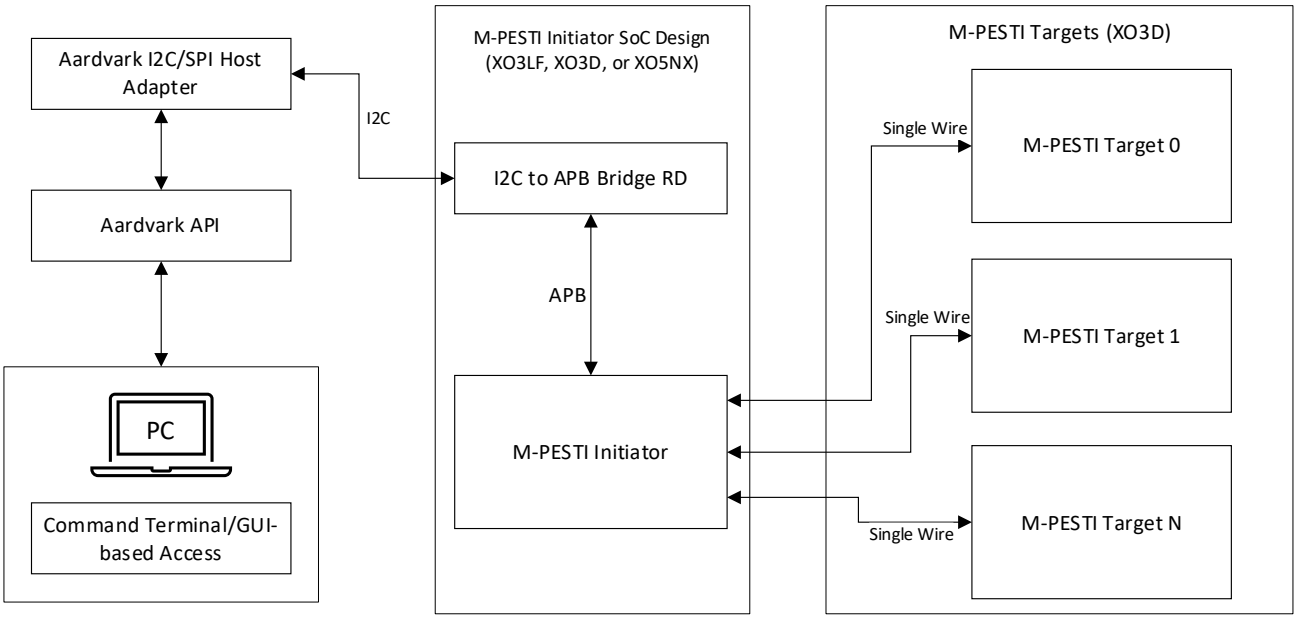

Figure 3.1. Reference Design Block Diagram

3.1. Design Components

The RISC-V Controlled M-PESTI Initiator reference design includes the following blocks:

- M-PESTI Initiator SoC design
  - M-PESTI IP core
  - I2C to APB Bridge reference design
- M-PESTI pseudo-target module with multiple emulated target devices
- PC-based Command Terminal/GUI-based access based on the included Aardvark Python(*.py*) reference code

3.1.1. M-PESTI Initiator SoC Design

The following figure shows that the M-PESTI Initiator IP core is instantiated in the SoC design and uses the I2C to APB Bridge reference design for IP register access to orchestrate the IP functionality. This Lattice Propel Builder SoC design can be used to generate a Diamond™ software project, which is then used to generate the programming bitstreams.

For more information, refer to the Lattice Propel 2025.2 Builder User Guide (FPGA-UG-022434). Pre-generated bitstreams are included in the reference design package, and regeneration is required only when design modifications are made.

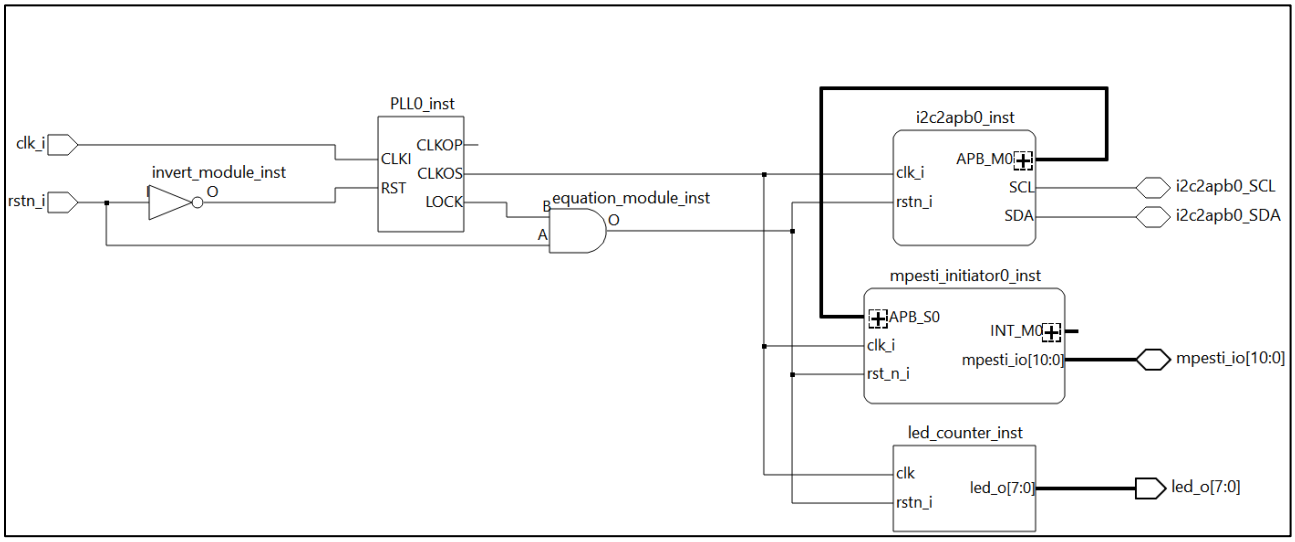

Figure 3.2. M-PESTI Initiator Reference Design Interface Diagram

3.1.2. M-PESTI Target Module

The M-PESTI pseudo-target device is programmed into a MachXO3D Breakout Board. Several instantiations of the M-PESTI target module are programmed onto a separate FPGA board to emulate multiple M-PESTI target devices. These modules are connected to the initiator through the 1-wire interface. This target module has an internal loopback mechanism that echoes the last Virtual Wire (VW) data byte it received from the previous transaction back to the initiator. For example, during transaction 1, the last data byte that it received from the initiator is 0xAA. During transaction 2, the target responds 0xAA back to the initiator.

3.1.3. Command Terminal/GUI-Based Access

To control the M-PESTI Initiator IP, the design uses either the Command Terminal or the GUI-based application, both written in Python (.py) to manage IP functionality and display messages on the PC.

The Python source codes are in **M-PESTI_Initiator_Reference_Design/Aardvark_Source_Code/python** folder. This folder contains:

- *mpesti_demo_cmd_terminal_source.py* — source code for the Command Terminal-based implementation
- *mpesti_demo_gui_source.py* — source code for GUI-based access.

Either option may be used, depending on preference.

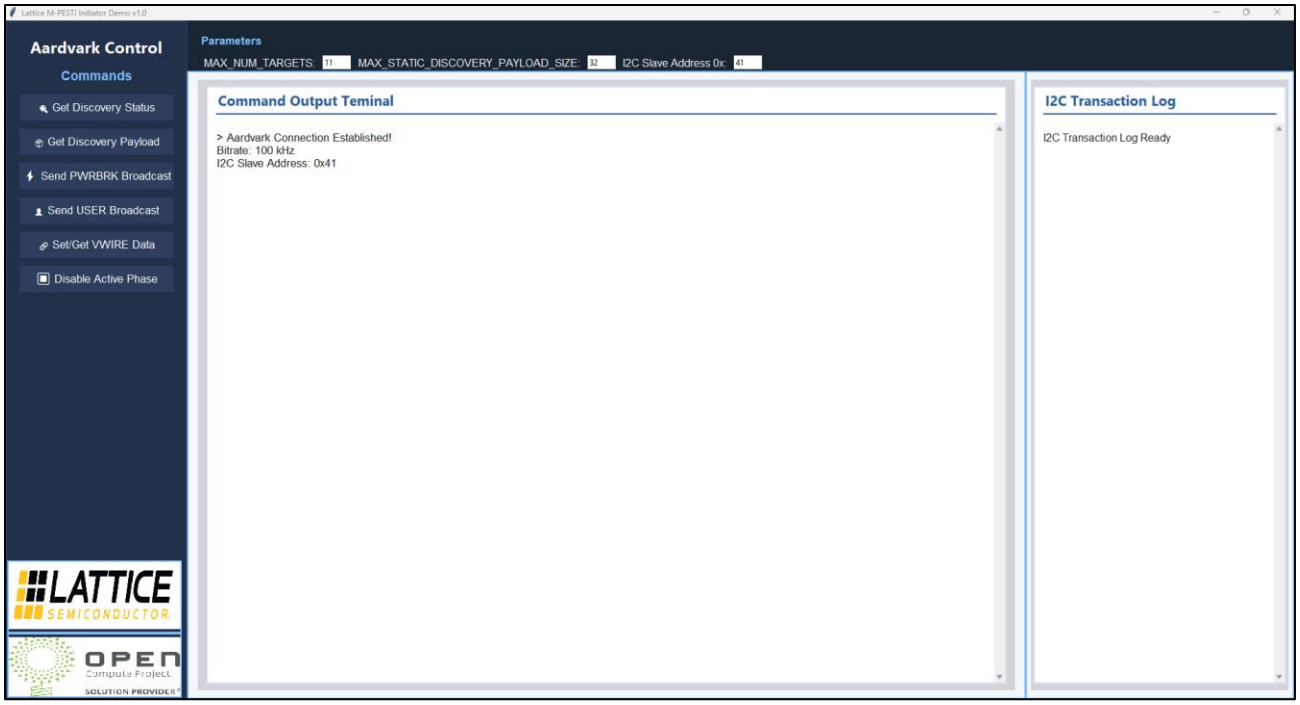

Figure 3.3. GUI-based Access

```
Aardvark Connection Port 0 Established!
Bitrate set to : 100 kHz
I2C Slave Address Set to : 0x41


Select an operation:
[1] Get Discovery Status
[2] Get Discovery Payload
[3] Send PWRBRK Broadcast
[4] Send USER Broadcast
[5] Set/Get VWIRE Data
[6] Disable Active Phase
[7] Exit/Close Aardvark Terminal
Enter choice [1-7]:
```

Figure 3.4. Command Terminal-based Access

3.2. M-PESTI Protocol Phases

3.2.1. Discovery Phase

After power-on, the M-PESTI target enters an initialization state (break event) that lasts for T_DBREAK_NS (50 µs). After this interval, the M-PESTI target releases the M-PESTI line (break release) and becomes ready to respond to the Discovery Payload Request Command. The following figure shows three targets undergoing the Discovery Phase in a round-robin fashion. The first 8-bit transaction on each of the targets is the Discovery Payload Request command (0x00) sent by the initiator.

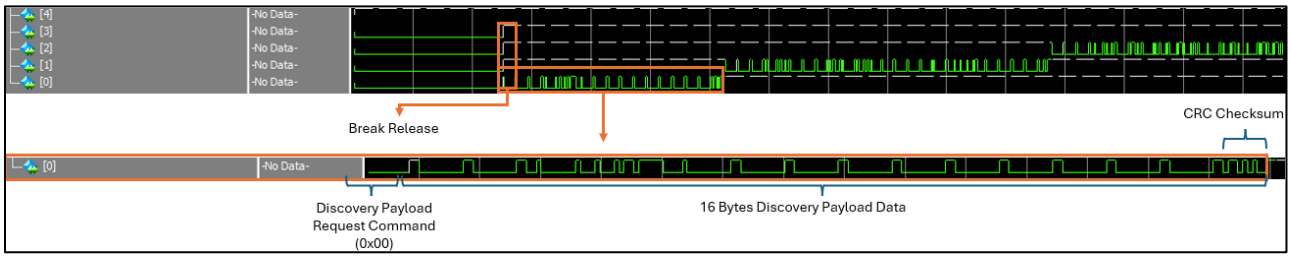

Figure 3.5. Discovery Phase

In the zoomed-out image of Target 0 above, the target responds with 16 discovery payload data bytes, which match the third header byte value (STATIC_PAYLOAD_SIZE[7:0] x 8) of the discovery payloads.

According to the M-PESTI specification, the STATIC_PAYLOAD_SIZE value represents the number of discovery payload data bytes divided by 8.

For example, STATIC_PAYLOAD_SIZE = 0x02h indicates the size as 2 × 8 = 16 discovery payload data bytes.

3.2.2. Active Phase

During the Active Phase, the initiator issues a Virtual Wire (VW) Exchange Request command (0x01) to the target and begins transmitting VW data. Next, the target responds by also transmitting VW data to the initiator. The number of VW output and VW input bytes is determined by the fourth byte of the target's discovery payload.

- The bits [3:0] of the fourth byte indicate the number of VW input bytes from initiator to target (NUM_VIRTUAL_WIRE_INPUT_BYTES).
- The bits [7:4] of the fourth byte indicate the number of VW output bytes from target to initiator (NUM_VIRTUAL_WIRE_OUTPUT_BYTES).

After completing the VW exchange on Target 0, the sequence is repeated for the other remaining targets in a round-robin fashion. The following figure shows the VW data exchange activity.

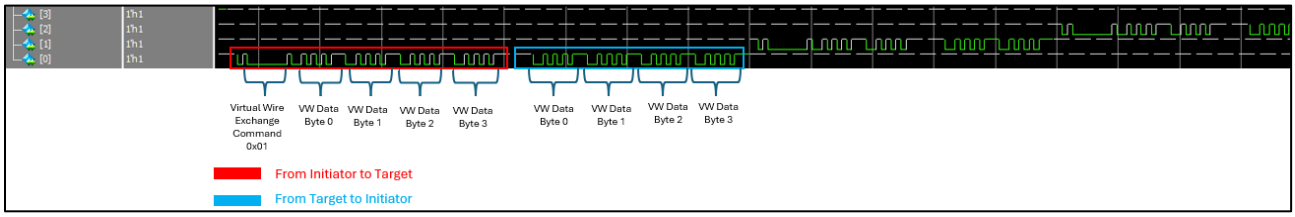

Figure 3.6. Active Phase Virtual Wire Exchange Data

3.2.3. Broadcast Transactions

The broadcast request can be triggered only during the Active Phase. If no VW exchange transaction is in progress, the broadcast command can be transmitted immediately.

However, if the broadcast request is triggered while VW exchange is in progress, the request is delayed until the current VW exchange transaction is complete.

This scenario is shown in the following figure, in which the Broadcast Request Command (0xFF) only happens at the M-PESTI line after the VW exchange completes.

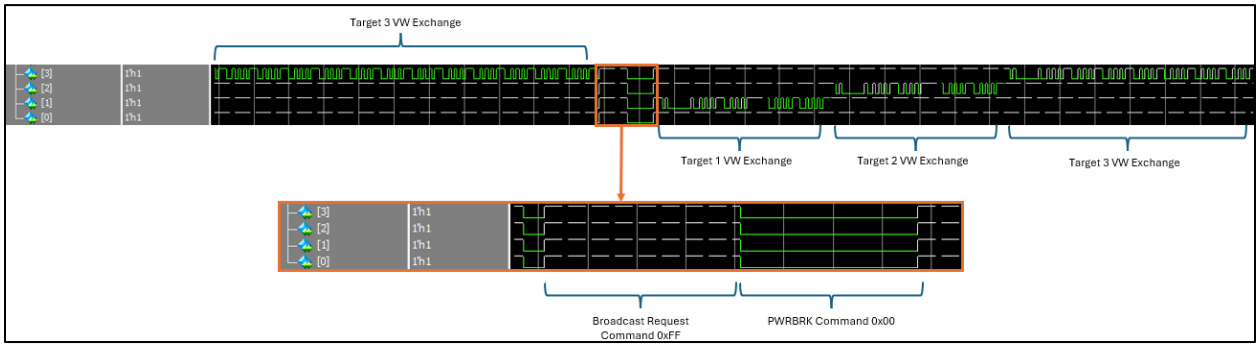

Figure 3.7. Broadcast Request

4. Pinout

4.1. M-PESTI Initiator Pinout

The following tables show the M-PESTI Initiator pinout for:

- MachXO3LF Starter Kit
- MachXO3D Breakout Board
- MachXO5-NX Development Board

Table 4.1. M-PESTI Initiator Pinout for MachXO3LF Starter Kit and MachXO3D Breakout Board

Name	Width	Direction	MachXO3LF/MachXO3D Ball	MachXO3LF Starter Kit/MachXO3D Breakout Board Component	Description
Clock and Reset Interface					
clk_i	1	input	C8	—	12 MHz clock input for MachXO3LF and MachXO3D.
rstn_i	1	input	B3	SW1	Active low reset signal.
M-PESTI Initiator Interface					
mpesti_io[0]	1	inout	D9	J3 (29)	M-PESTI line. Each pin can be left floating or connected to an M-PESTI target.
mpesti_io[1]	1	inout	A10	J3 (27)	
mpesti_io[2]	1	inout	E6	J3 (26)	
mpesti_io[3]	1	inout	C5	J3 (25)	
mpesti_io[4]	1	inout	D7	J3 (19)	
mpesti_io[5]	1	inout	E7	J3 (17)	
mpesti_io[6]	1	inout	E8	J3 (15)	
mpesti_io[7]	1	inout	F9	J3 (13)	
mpesti_io[8]	1	inout	E10	J3 (9)	
mpesti_io[9]	1	inout	C12	J3 (7)	
mpesti_io[10]	1	inout	F8	J3 (5)	
i2c2apb0_SCL	1	input	B4	J3 (37)	I2C interface serial clock line.
i2c2apb0_SDA	1	inout	A4	J3 (38)	I2C interface serial data line.
Others					
led_o[0]	1	output	H11	LED D9	LED counter indicates that the system clock is present and the PLL has achieved locked state.
led_o[1]	1	output	J13	LED D8	
led_o[2]	1	output	J11	LED D7	
led_o[3]	1	output	L12	LED D6	
led_o[4]	1	output	K11	LED D5	
led_o[5]	1	output	L13	LED D4	
led_o[6]	1	output	N15	LED D3	
led_o[7]	1	output	P16	LED D2	

Table 4.2. M-PESTI Initiator Pinout for MachXO5-NX Development Board

Name	Width	Direction	MachXO5-NX Ball	MachXO5-NX Development Board Component	Description
Clock and Reset Interface					
clk_i	1	input	V1	—	125 MHz clock input for MachXO5-NX.
rstn_i	1	input	G20	SW5	Active low reset signal.
M-PESTI Initiator Interface					
mpesti_io[0]	1	inout	E4	J9 (4)	M-PESTI line. Each pin can be left floating or connected to an M-PESTI target.
mpesti_io[1]	1	inout	C2	J9 (6)	
mpesti_io[2]	1	inout	E5	J9 (8)	
mpesti_io[3]	1	inout	C5	J9 (10)	
mpesti_io[4]	1	inout	A2	J9 (12)	
mpesti_io[5]	1	inout	A3	J9 (14)	
mpesti_io[6]	1	inout	D5	J9 (16)	
mpesti_io[7]	1	inout	B5	J9 (18)	
mpesti_io[8]	1	inout	D3	J9 (3)	
mpesti_io[9]	1	inout	C3	J9 (5)	
mpesti_io[10]	1	inout	A4	J9 (7)	
i2c2apb0_SCL	1	input	E11	J16(1)	I2C interface serial clock line
i2c2apb0_SDA	1	inout	D13	J16(2)	I2C interface serial data line
Others					
led_o[0]	1	output	R3	LED D1	LED counter indicates that the system clock is present and the PLL has achieved locked state.
led_o[1]	1	output	R2	LED D2	
led_o[2]	1	output	R1	LED D3	
led_o[3]	1	output	P7	LED D4	
led_o[4]	1	output	H12	LED D5	
led_o[5]	1	output	H11	LED D6	
led_o[6]	1	output	G13	LED D7	
led_o[7]	1	output	G12	LED D8	

4.2. M-PESTI Target Pinout

The following table shows the M-PESTI Target pinout for MachXO3D Breakout Board.

Table 4.3. M-PESTI Target Pinout

Name	Width	Direction	MachXO3D Ball	MachXO3D Breakout Board Component	Description
Clock and Reset Interface					
clk_i	1	input	C8	—	12 MHz clock input.
reset_n	1	input	B3	SW1	Active low reset signal. SW1 tact switch.
M-PESTI Target Interface					
mpesti_t[0]	1	inout	D1	J8 (30)	M-PESTI line. Each pin can be left floating or connected to an M-PESTI initiator.
mpesti_t[1]	1	inout	E1	J8 (28)	
mpesti_t[2]	1	inout	F1	J8 (26)	
mpesti_t[3]	1	inout	G1	J8 (24)	
Others					
led_o	1	output	H11	LED D9	LED blink indicator.
loopback_en_i	1	input	P2	SW2 (DIP_SW1)	Active high signal. Enables the target to echo back the last VW data byte value it received from the previous transaction to the current transaction.

5. Implementing the Reference Design on Board

This reference design requires two boards:

- M-PESTI Initiator SoC Design: MachXO3LF Starter Kit, MachXO3D Breakout Board, or MachXO5-NX Development Board
- M-PESTI target module: MachXO3D Breakout Board

Jumper wire connections are required to bridge the two boards.

In Figure 5.1, the board on the left is the MachXO3LF Starter Kit, acting as the M-PESTI Initiator, while the board on the right is the MachXO3D Breakout Board, acting as the emulated M-PESTI targets.

Note: This figure is also applicable when using the MachXO3D Breakout Board as the M-PESTI Initiator since it has the same pinout as the MachXO3LF Starter Kit in the design.

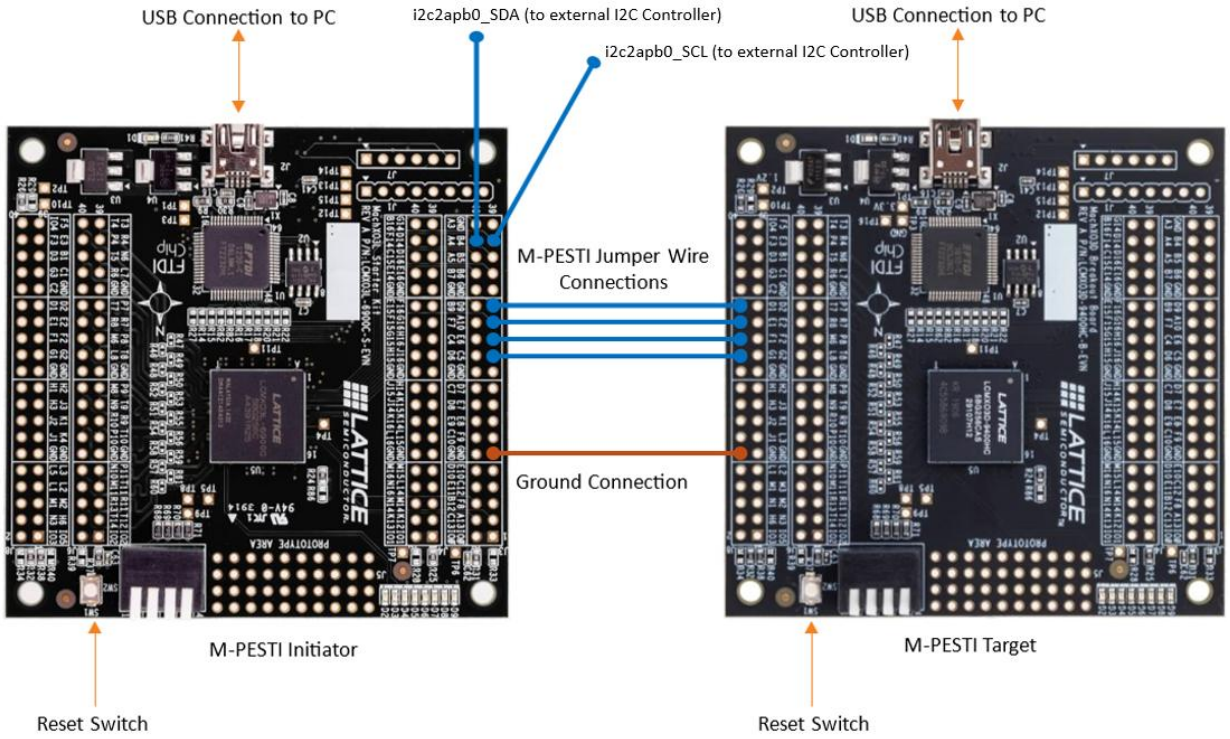

Figure 5.1. Hardware Connections between MachXO3LF and MachXO3D Devices

Figure 5.2 shows wherein the initiator is using MachXO5-NX Development Board.

Although only one target board, the design programmed into the board emulates four independent target devices, connected through each of the jumper wire connections. It is also good practice to have a ground connection between the two boards.

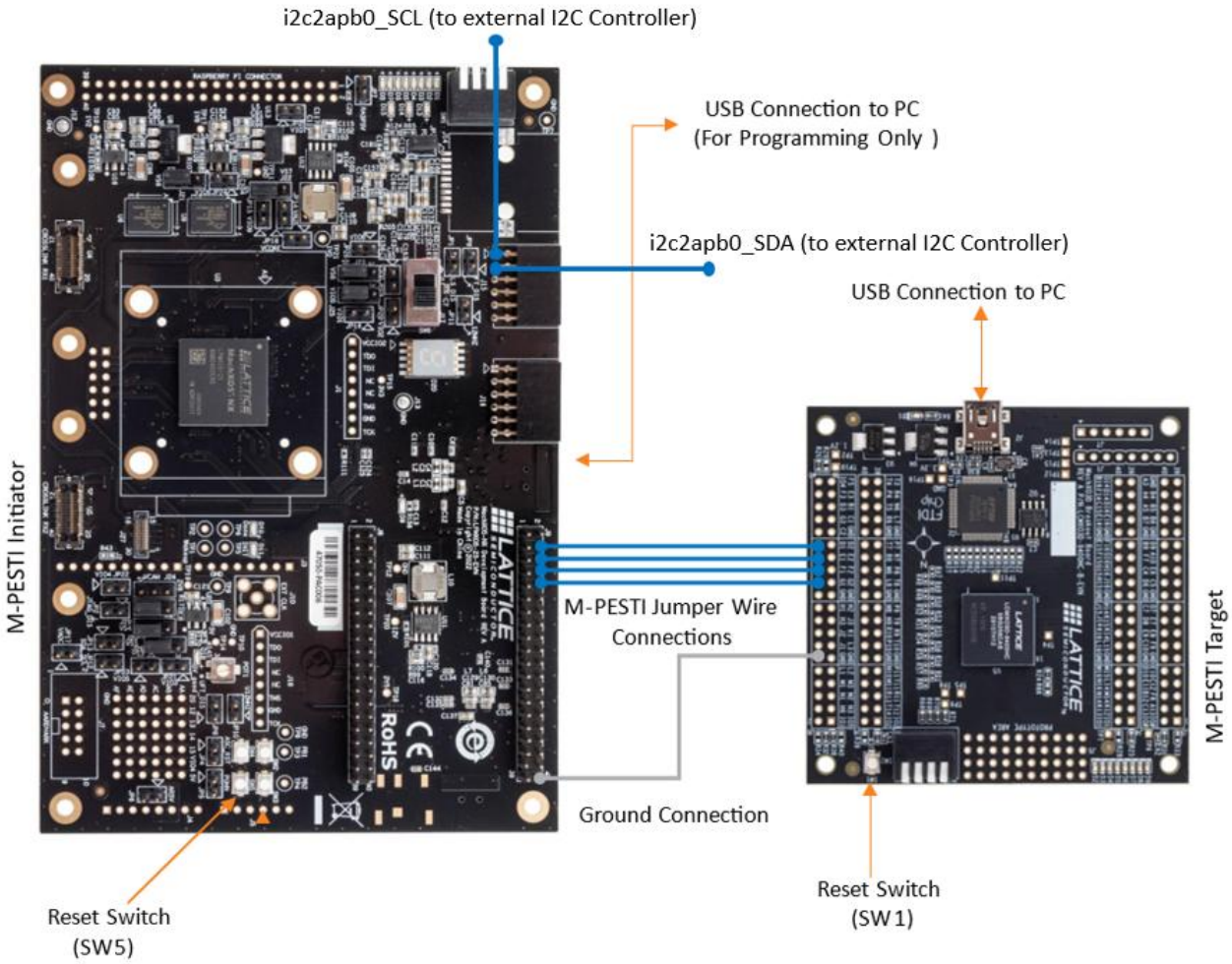

Figure 5.2. Hardware Connections between MachXO5-NX and MachXO3D Devices

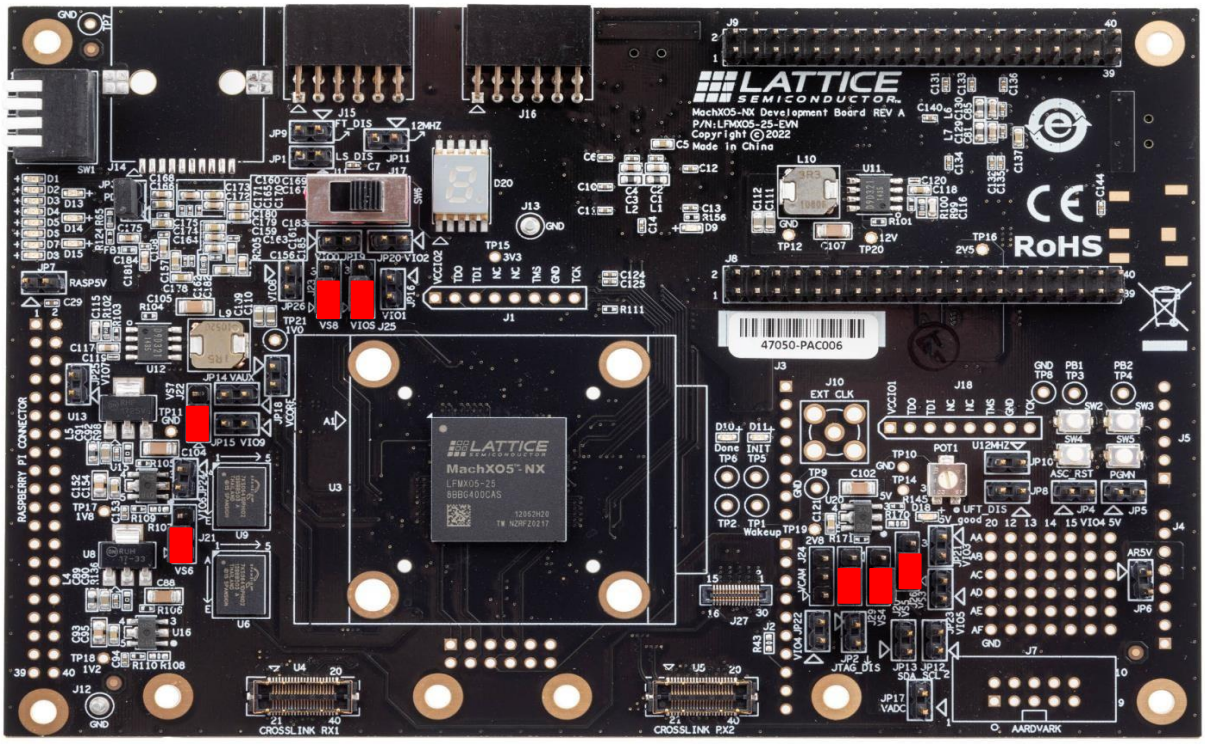

Figure 5.3. Jumper Settings for MachXO5-NX Development Board

5.1. Programming the M-PESTI Initiator SoC Design to the MachXO3LF Starter Kit or MachXO3D Breakout Board

To program the M-PESTI Initiator design onto the MachXO3LF Starter Kit, follow the steps below. Note that these steps are also applicable for programming the MachXO3D Breakout Board as an M-PESTI initiator. Only the programming file and the target device need to be changed. The programming files are in the Bitstream folder.

1. Connect the MachXO3LF Starter Kit to the USB port.
2. Launch the Diamond programmer. Select **Create a new blank project** as shown in the following figure.

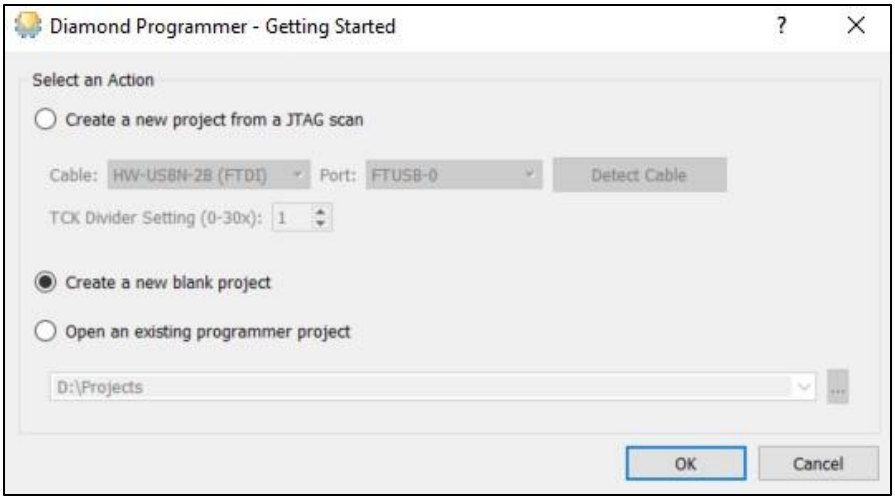

Figure 5.4. Create a New Blank Project

3. Click the JTAG Scan button to detect the LCMXO3LF-6900C device found on the MachXO3LF Starter Kit board.

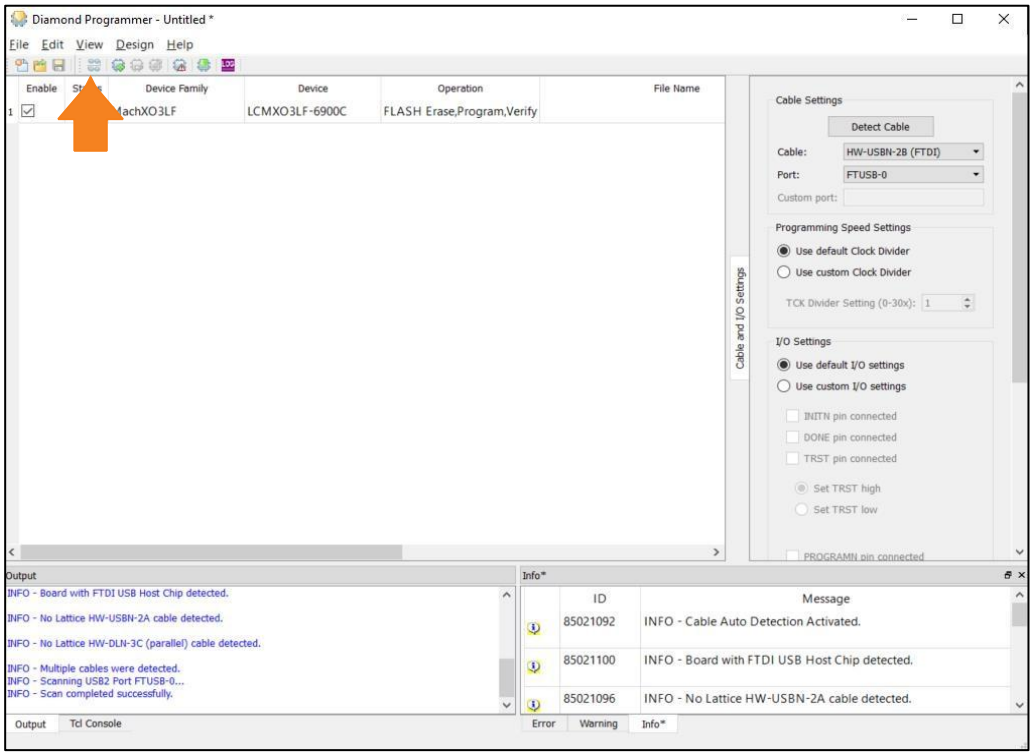

Figure 5.5. JTAG Scan

4. Right-click on the device and select **Device Properties** as shown in the following figure.

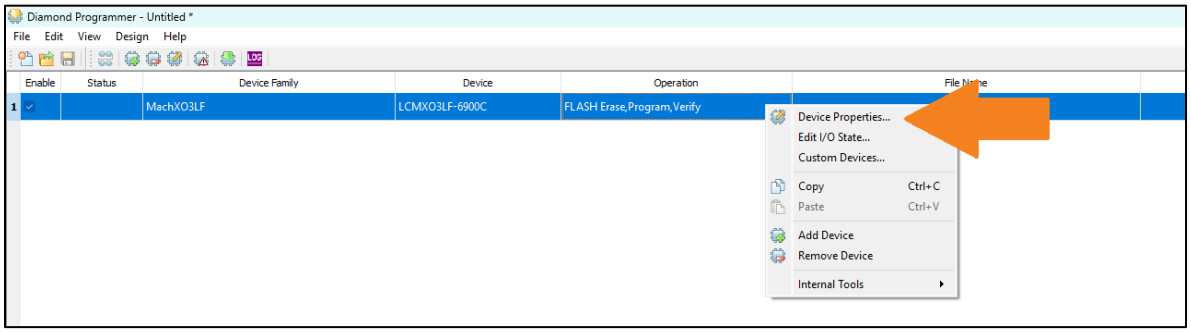

Figure 5.6. Diamond Programmer - Device Properties

5. Do the following settings as shown in the following figure and click **OK**.
   - Access Mode: **Flash Programming Mode**
   - Operation: **FLASH Erase, Program, Verify**
   - Programming File:
     MachXO3LF: ***XO3LF_MPESTI_Initiator_Demo_impl1.jed***
     MachXO3D: ***XO3D_MPESTI_Initiator_Demo_impl1_a.jed***

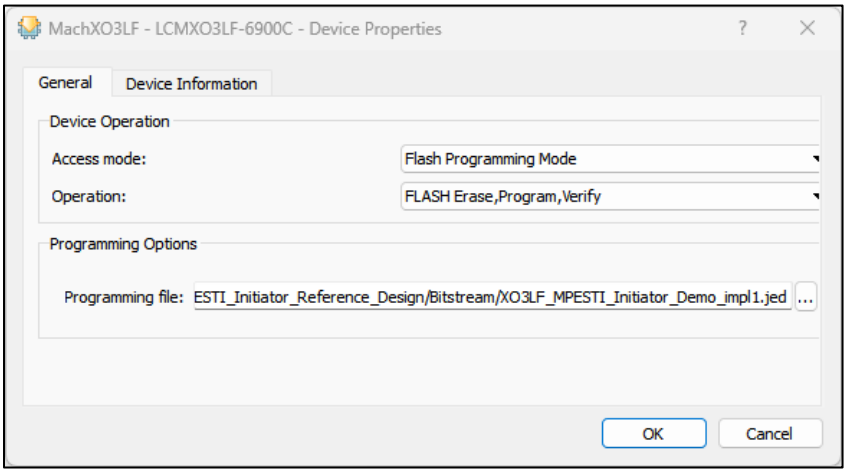

Figure 5.7. MachXO3LF Device Properties Settings

6. Click the Program button as shown in the following figure. It is recommended to set the TCK Divider Setting to 3 or above to have a slower and more stable programming clock frequency.

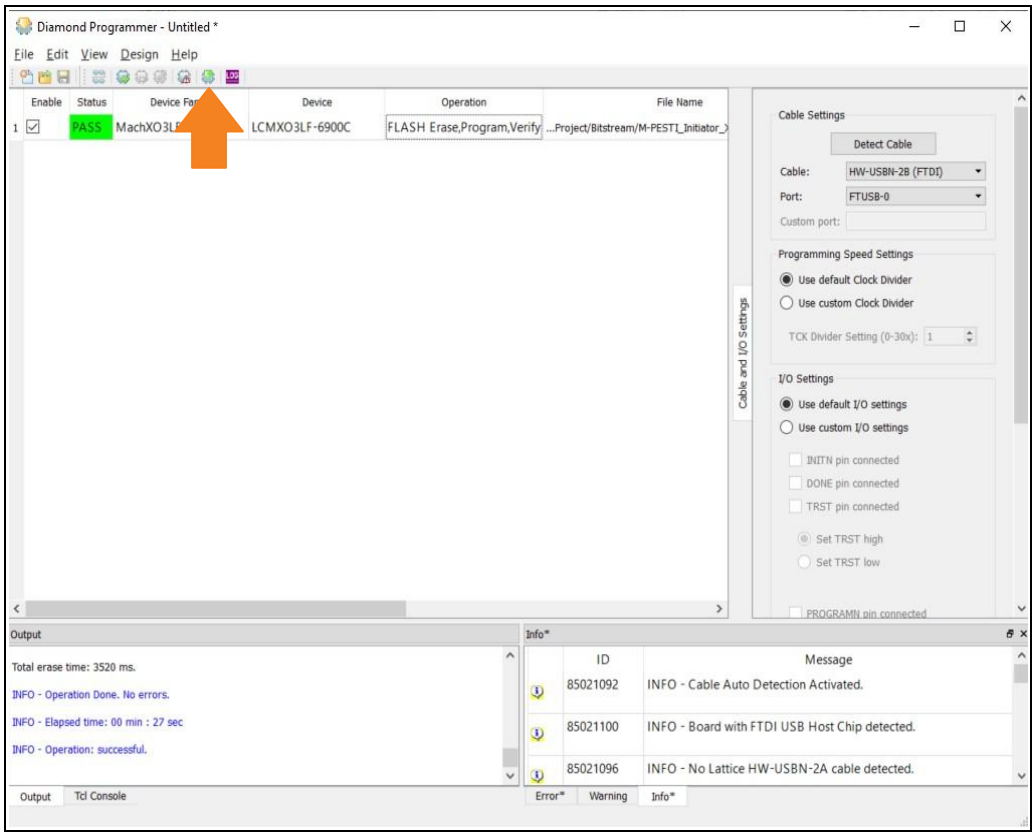

Figure 5.8. Program Button

7. When programming is successful, the output console displays the message shown in the following figure.

Figure 5.9. Operation Successful

5.2. Programming the M-PESTI Initiator SoC Design to the MachXO5-NX Development Board

To program the M-PESTI Initiator design to the MachXO5-NX Development Board, follow these steps:

1. Connect the MachXO5-NX Development Board to the USB port.
2. Launch the Radiant Programmer. Select Create a new blank project as shown in the following figure.

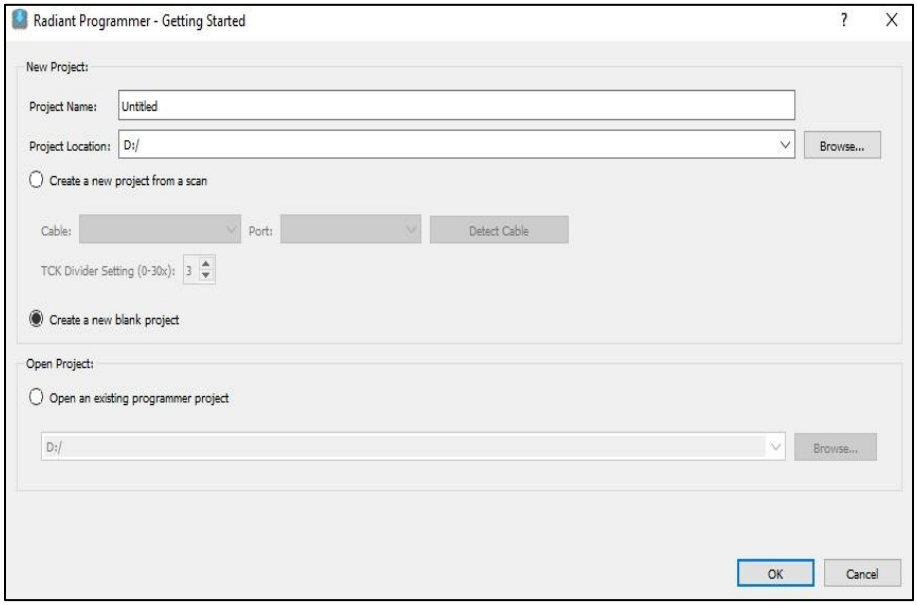

Figure 5.10. Create a New Blank Project (MachXO5-NX)

3. Click the JTAG Scan button to detect the LFMXO5-25 device found on the MachXO5-NX Development Board.

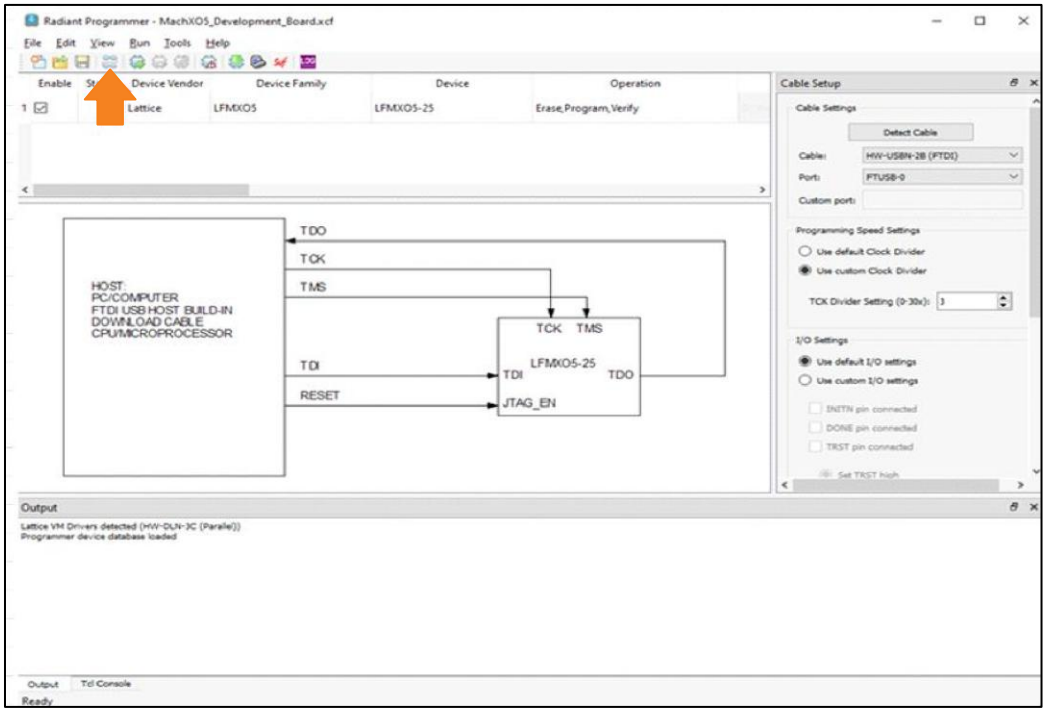

Figure 5.11. JTAG Scan (MachXO5-NX)

4. Right-click on the device and select **Device Properties** as shown in the following figure.

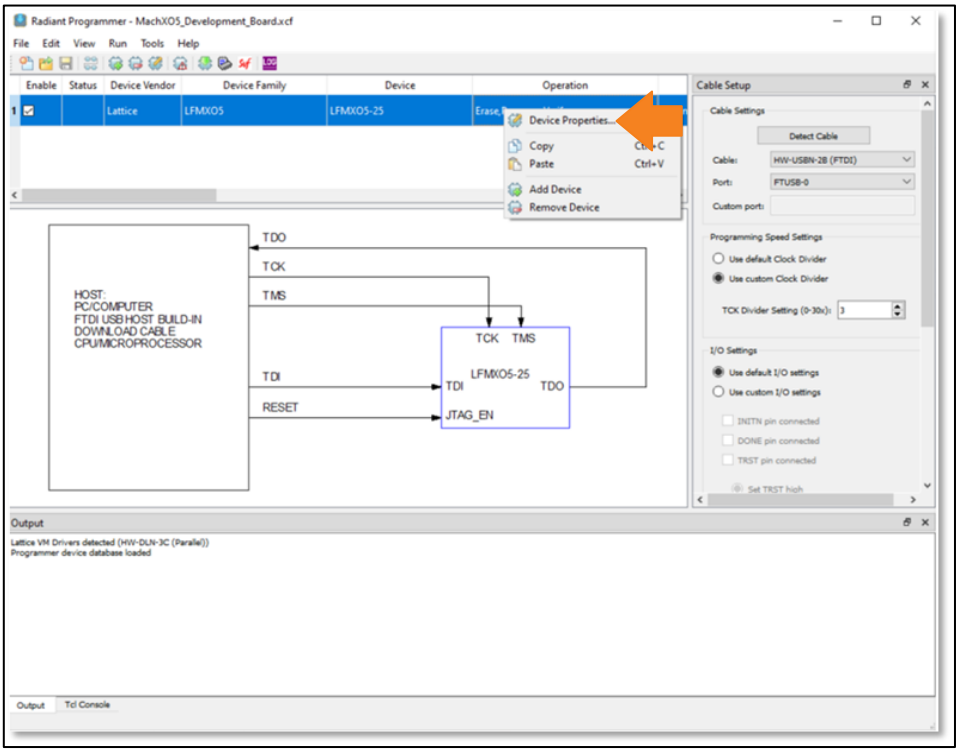

Figure 5.12. Radiant Programmer - Device Properties (MachXO5-NX)

5. Do the following settings as shown in the following figure and click **OK**.
   - Target Memory: **Flash Configuration Memory**
   - Port Interface: **JTAG**
   - Access Mode: **Direct FLASH Programming**
   - Operation: **Erase, Program, Verify, Refresh**
   - CFG0 Programming Options: **Checked**
   - Programming File: ***XO5NX_MPESTI_Initiator_Demo_impl_1_0.jed***

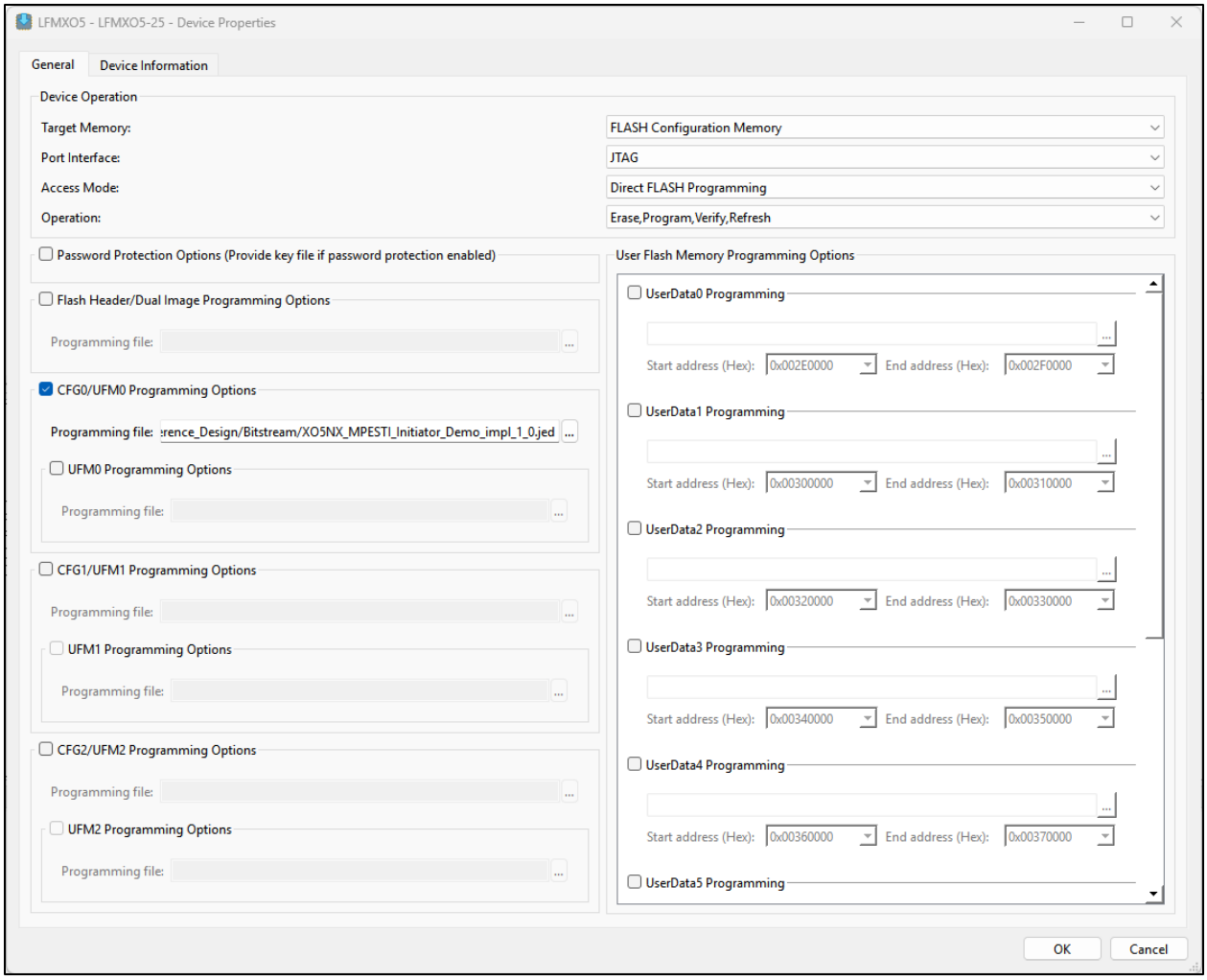

Figure 5.13. MachXO5-NX Device Properties (MachXO5-NX)

6. Click the Program button as shown in the following figure.

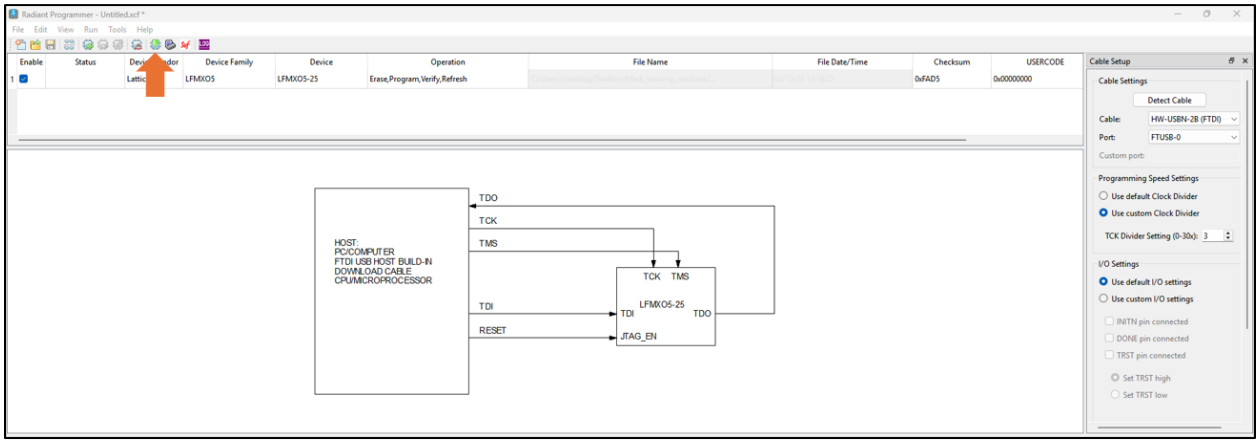

Figure 5.14. Program Button

7. When programming is successful, the output console displays the message shown in the following figure.

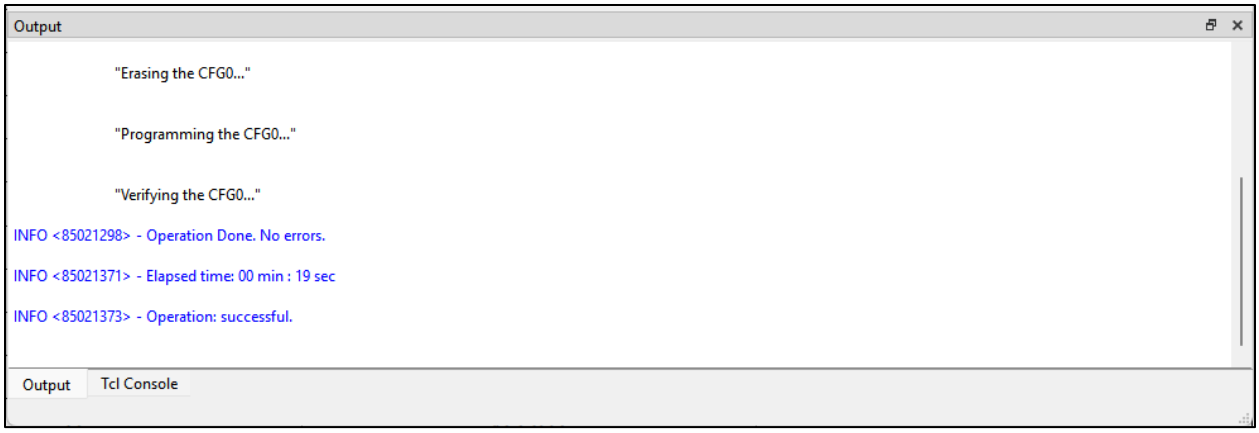

Figure 5.15. Operation Successful

5.3. Programming the M-PESTI Target Module to the MachXO3D Breakout Board

To program the M-PESTI target module to the MachXO3D Breakout Board, follow these steps:

1. Connect the MachXO3D Breakout Board to the USB port.
2. Launch the Diamond Programmer. Create a **new blank project**.
3. Click the **JTAG Scan** button to detect the LCMXO3D-9400HC device found on the MachXO3D Breakout Board.

Figure 5.16. LCMXO3D-9400HC Device Selection

4. Do the following settings and click **OK**:
   - Access Mode: **Flash Programming Mode**
   - Port Interface: **JTAG Interface**
   - Operation: **FLASH Erase, Program, Verify**
   - CFG0 Programming Options: **Checked**
   - Programming File: ***XO3D_Target_impl1_a.jed***

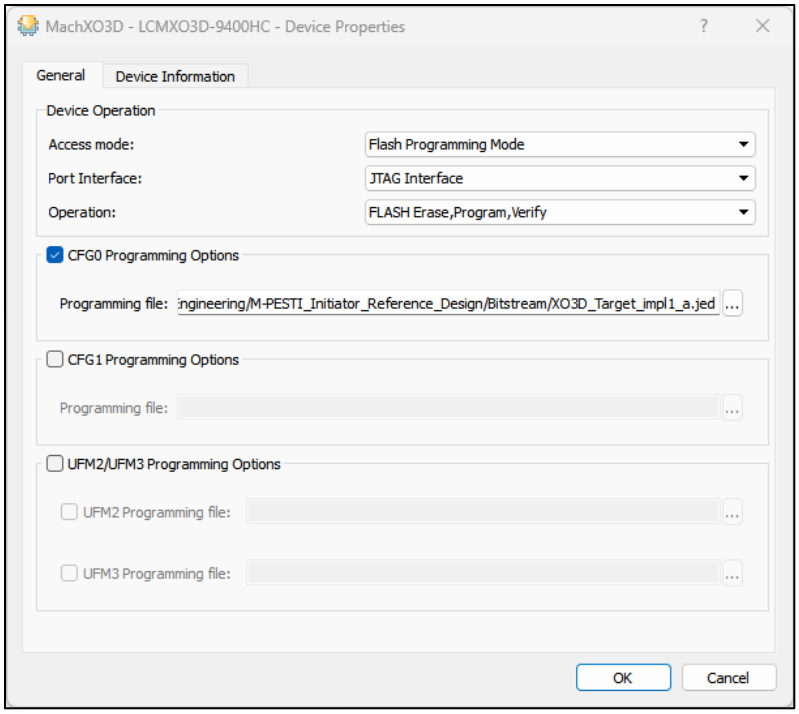

Figure 5.17. MachXO3D Device Properties Settings

6. IP Demonstration

6.1. Enabling the Aardvark I2C/SPI Host Adapter as I2C Controller

6.1.1. I2C Interface Board Connect

To enable access to the M-PESTI Initiator IP via I2C, you must connect the Aardvark I2C/SPI Host Adapter to the M-PESTI Initiator Board as shown below.

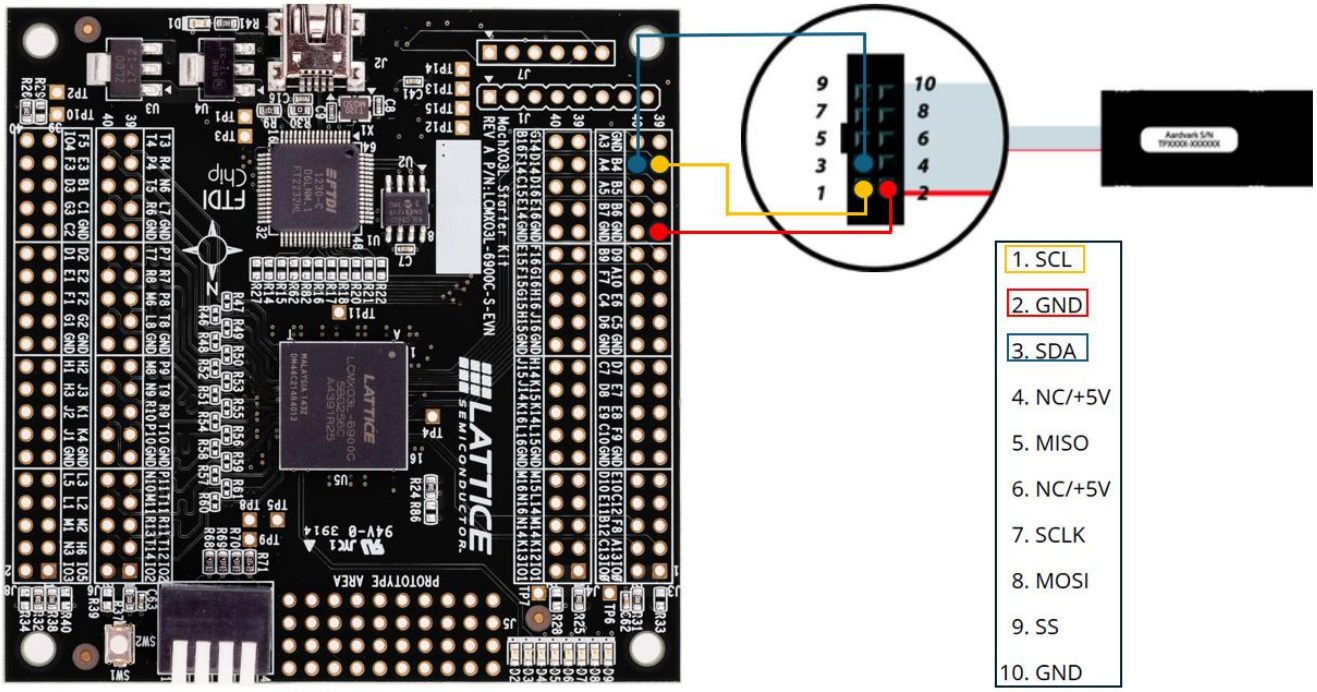

Figure 6.1. I2C Interface Connection

6.1.2. Running the Reference Python Code

The reference Python code drives the I2C interface to gain access to and control of the M-PESTI Initiator IP registers.

Do not connect the USB port of the Aardvark device to the PC immediately after connecting the I2C interface (Figure 6.1) between the M-PESTI initiator and the Aardvark device. The following steps must be completed first:

1. Download Python version 3.13.3 from www.python.org/ftp/python/3.13.3/python-3.13.3-amd64.exe and install it. For convenience, the installer is also readily available in the **Installers** folder included in this reference design package. When prompted, select **Install Now**.

Figure 6.2. Python 3.13.3 Installation Prompt

2. Download the latest Aardvark I2C/SPI Host Adapter latest USB drivers from www.totalphase.com/products/usb-drivers/. For convenience, the installer is also readily available in the **Installers** folder included in this reference design package. Read and follow the instructions in the ***README.txt*** located in the file path ***installer/total_phase_usb-drivers/TotalPhaseUSB-v4.0.0/windows*** on how to install the USB drivers.

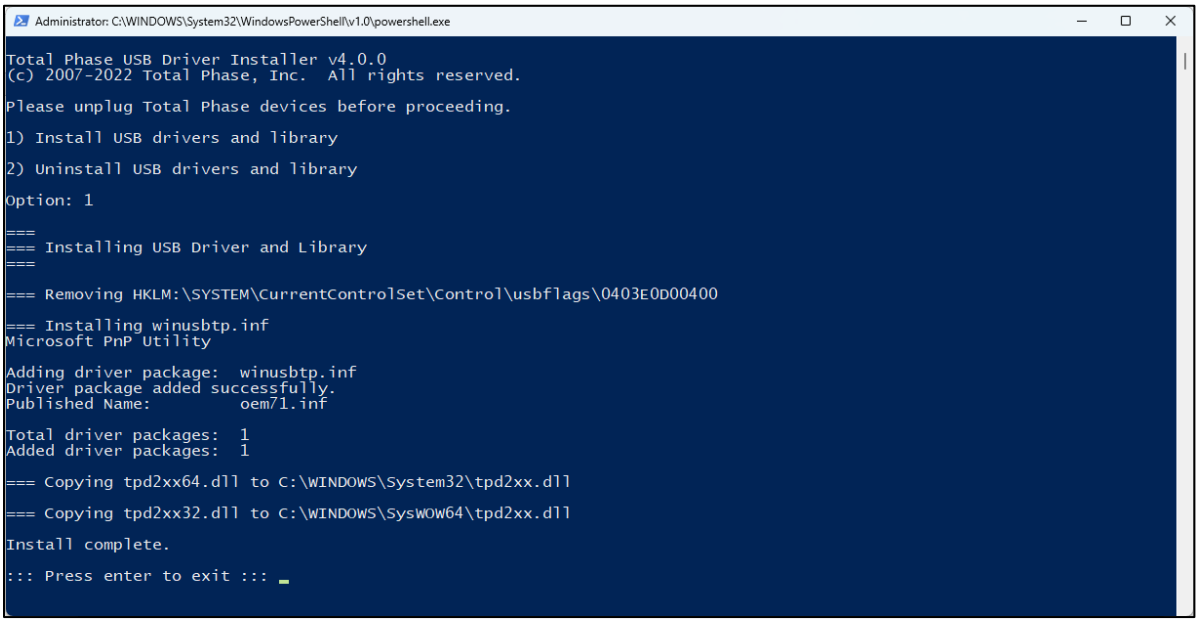

Figure 6.3. Aardvark USB Drivers Successfully Installed

3. Continue to this step only after you completed steps 1 and 2. Connect the USB port of the Aardvark I2C/SPI Host Adapter to your PC.
4. This step assumes that the following actions have already been completed successfully:

- Programmed the M-PESTI Initiator board with the bitstream described in sections 5.1 or 5.2, as applicable. The LEDs should blink at approximately 1-second intervals, indicating that the clock is present, the PLL has achieved a locked state, and the bitstream is running
- Programmed the M-PESTI target board (MachXO3D Breakout Board) with the bitstream described in section 5.3. The LED D9 should be blinking, indicating that the clock is present and the bitstream is running.
- The M-PESTI Initiator board and the target module board are connected as described in Figure 5.1 or Figure 5.2 as applicable
- The Aardvark I2C/SPI Host Adapter is connected to the I2C interface of the M-PESTI Initiator board, as described in Figure 6.1
- The USB port of the Aardvark I2C/SPI Host Adapter is connected to the PC

Launch Python IDLE 3.13.3, then select **File > Open >** choose either ***mpesti_demo_gui_source.py*** for GUI-based access or ***mpesti_demo_cmd_terminal_source.py*** for Command Terminal-based access. Both scripts are in the ***M-PESTI_Initiator_Reference_Design/Aardvark_Source_Code/python*** folder.

Note: Only one instance of the application can run at a time. Either the Terminal-based or the GUI-based reference code may be used, depending on preference. The GUI-based reference code includes a feature that logs the I2C transactions, whereas the Terminal-based reference application does not.

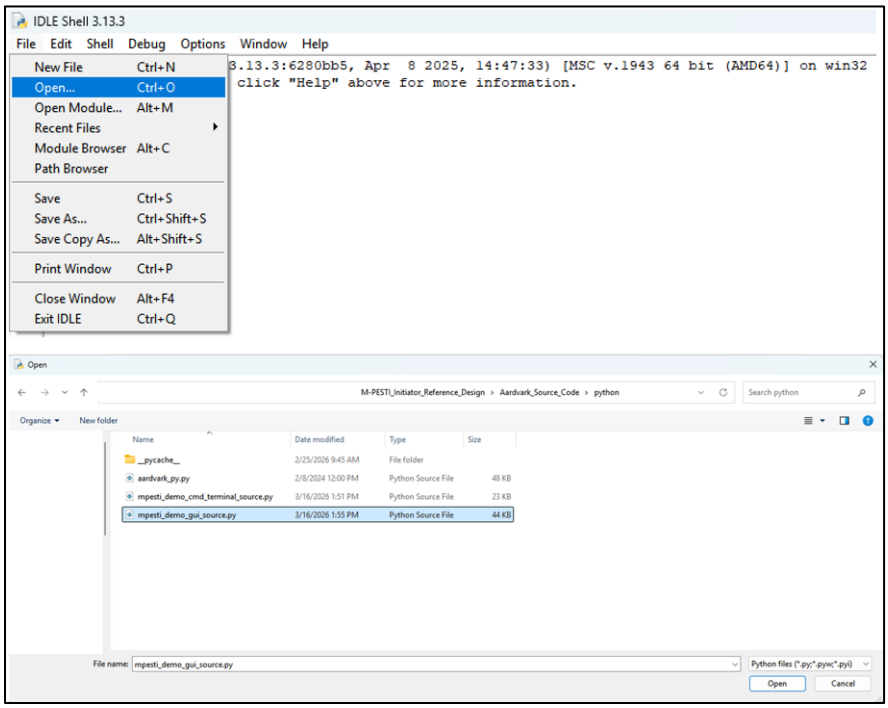

Figure 6.4. Opening Reference Python Code in Python IDLE

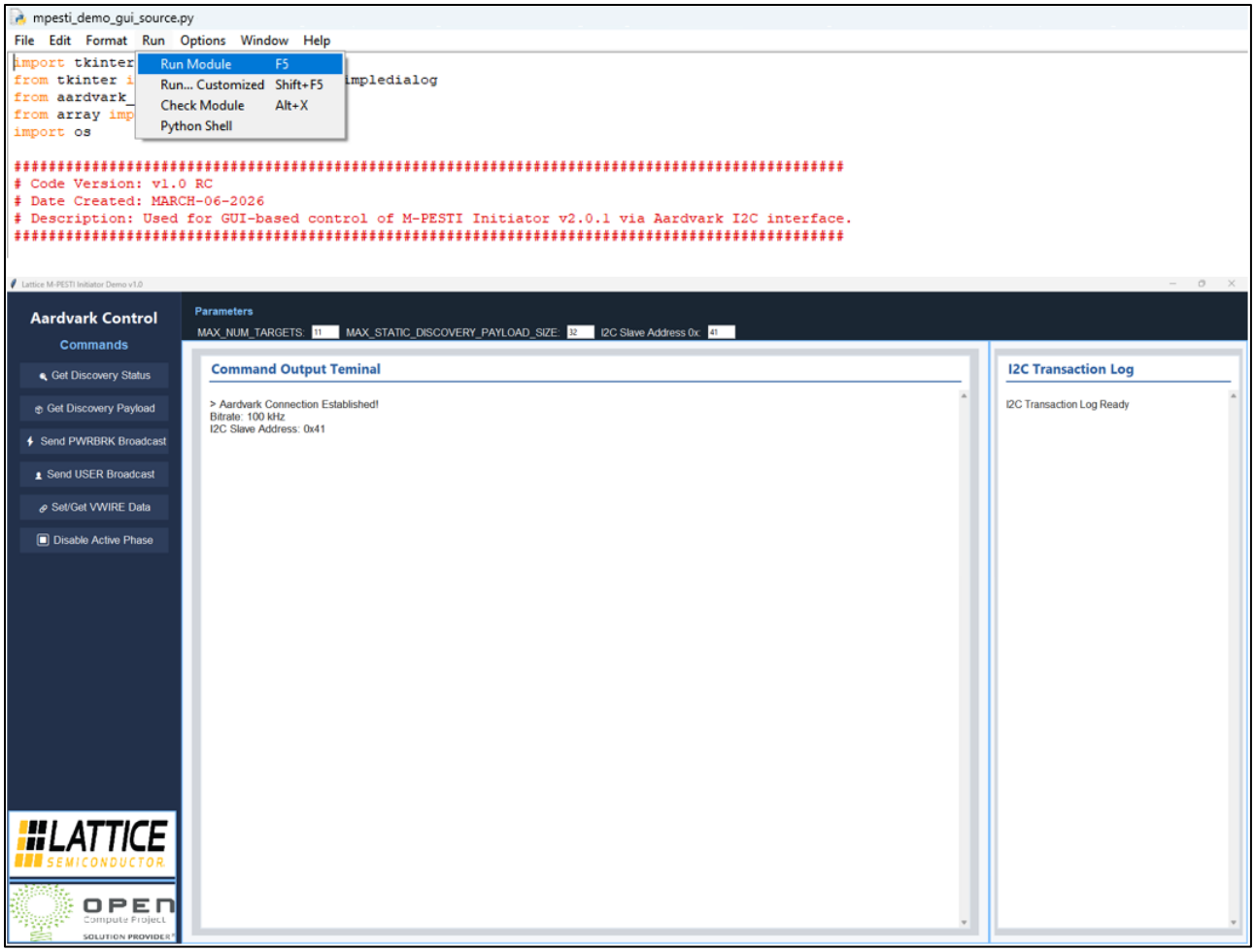

Figure 6.5. Successful Launch of the GUI-based Reference Python Code

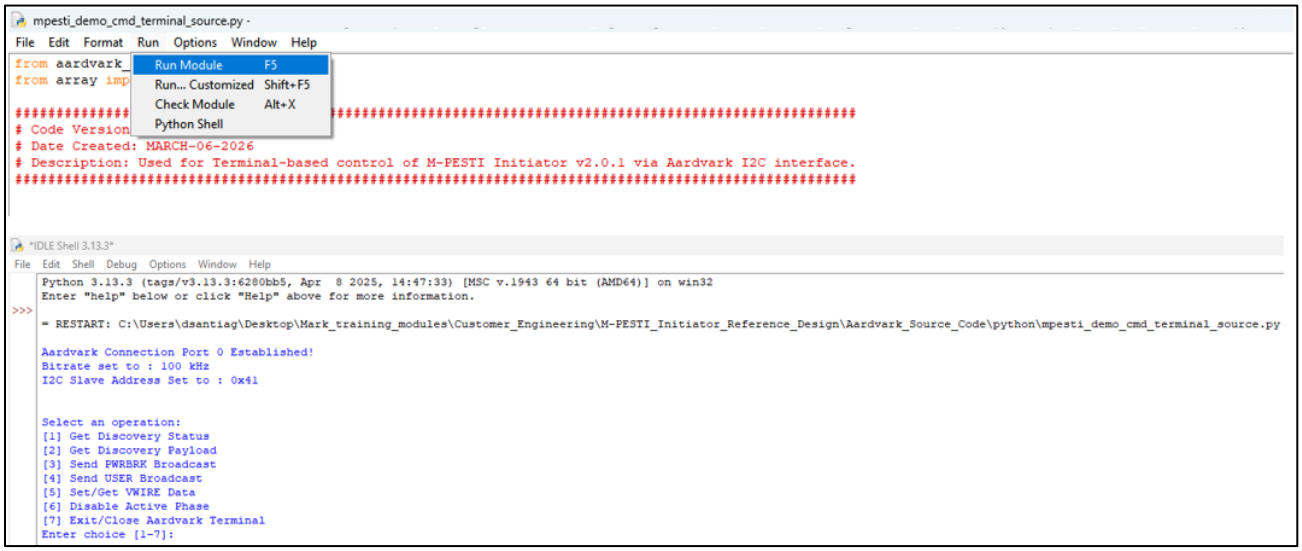

Figure 6.6. Successful Launch of the Terminal-based Reference Python Code

6.2. Usage Example

When the I2C connection is established, a selection menu and a success message appear on the screen. This menu provides access to various M-PESTI functions by selecting a numbered option and following the instructions for the selected item.

To port the I2C sequences to another external I2C controller or driver, you have an option to either probe the I2C transactions taking place in the I2C bus or refer to the I2C logs. Note the following information about the GUI-based access parameters:

- **MAX_NUM_TARGETS** – This setting must match the *MPESTI Initiator Maximum Number of Target Devices* attribute in the IP GUI
- **MAX_STATIC_DISCOVERY_PAYLOAD_SIZE** – This setting must match the *MPESTI Initiator Maximum Static Discovery Payload Size* attribute in the IP GUI
- **I2C Slave Address** – This setting must match the *I2C to APB Bridge RD 7-bit Slave Address* attribute in the GUI

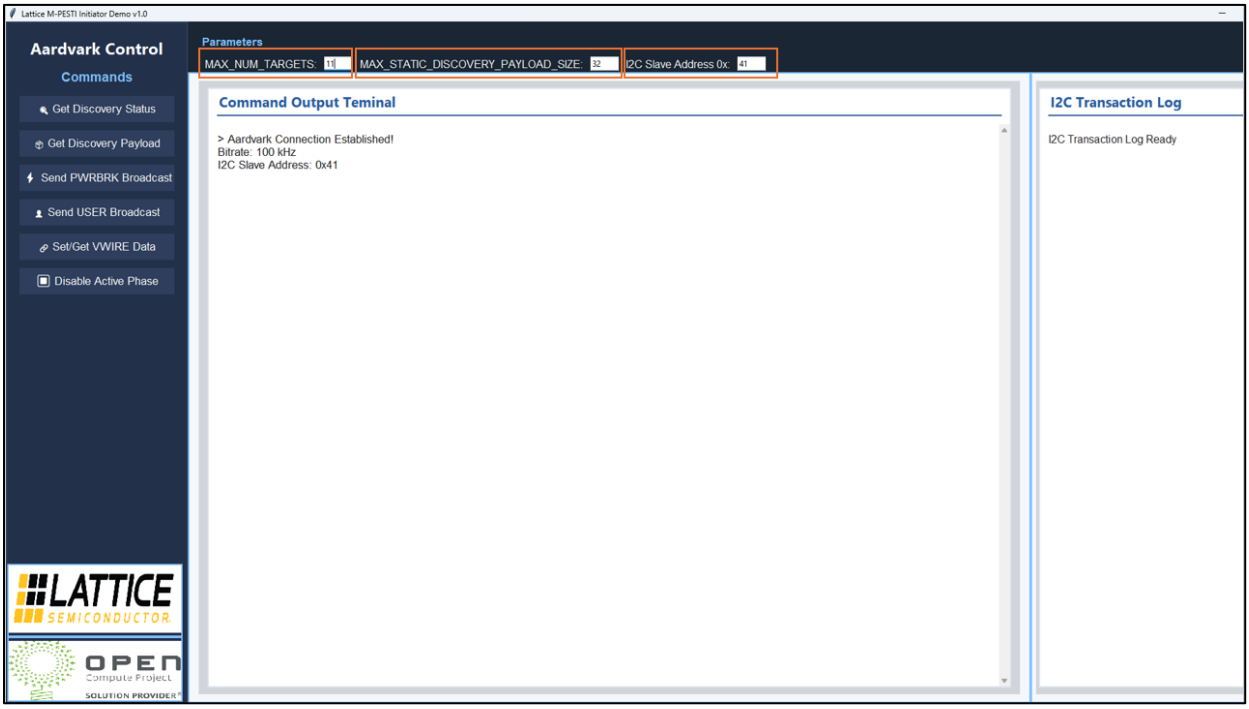

Figure 6.7. GUI-based Access Parameters

6.2.1. Get Discovery Status

Because the Discovery Phase Enable attribute in the GUI is enabled by default, the M-PESTI Initiator IP autonomously sends the Discovery Phase command (0x00) to any M-PESTI ready target that has provided a break release. Note that the reference code continues to follow the Payload Format defined in Specification 1.0, because Specification 1.2 does not yet fully define about the number of supported Virtual Wires. This menu selection allows the Discovery Status register to be read for each M-PESTI target. Each target may report one of the following statuses:

- 0b00 - Absent
- 0b01 - Present
- 0b10 - M-PESTI is present with good payload
- 0b11 – M-PESTI is present without successful payload yet

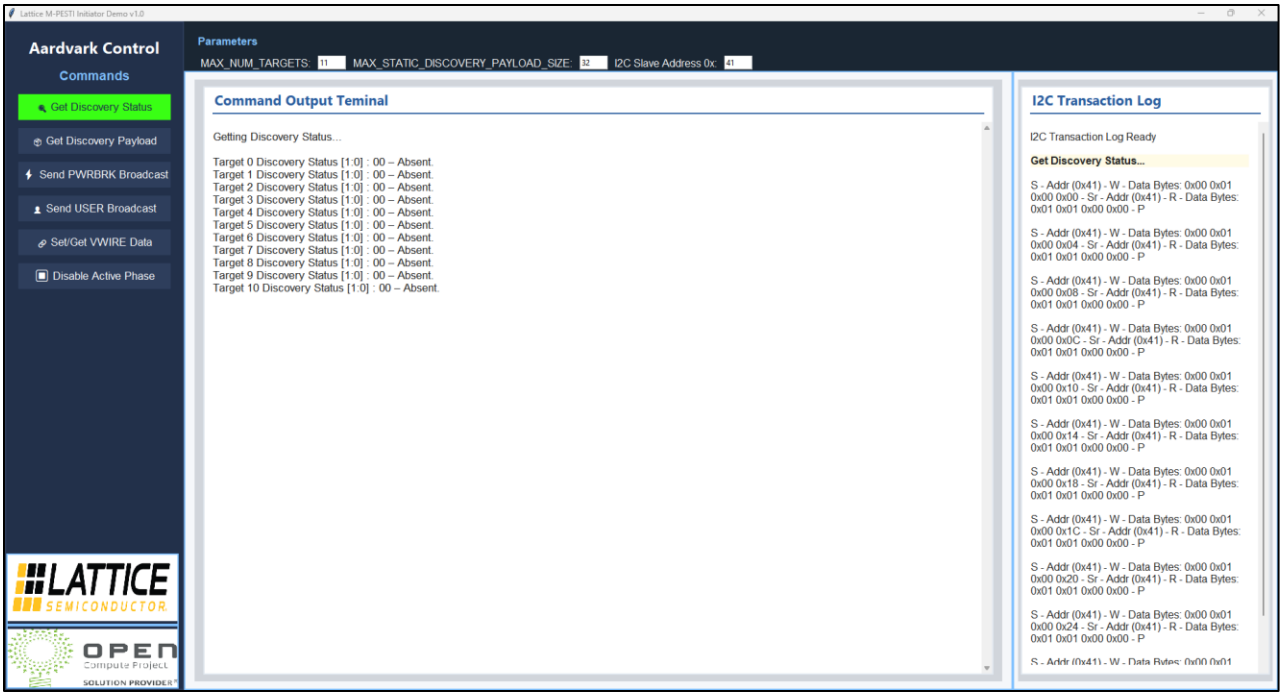

Figure 6.8. Absent Discovery Status

If the Absent status is displayed, this indicates that the M-PESTI Initiator is still waiting for a break release from the M-PESTI targets. A break release can be issued by pressing and releasing SW1 on the M-PESTI target board (MachXO3D Breakout Board). The figure below shows that when a break release is detected, the M-PESTI Initiator autonomously initiates the Discovery Phase and parses the discovery payloads of the targets.

Figure 6.9. Discovery Phase Hardware Waveform

After a break release is successfully issued, another Get Discovery Status can be performed. The figure below shows that the targets 0, 1, 2, and 3 HAVE been successfully discovered and the Discovery Payload has been read and stored by the M-PESTI Initiator. The other targets are not connected by default and therefore expected to be shown as Absent.

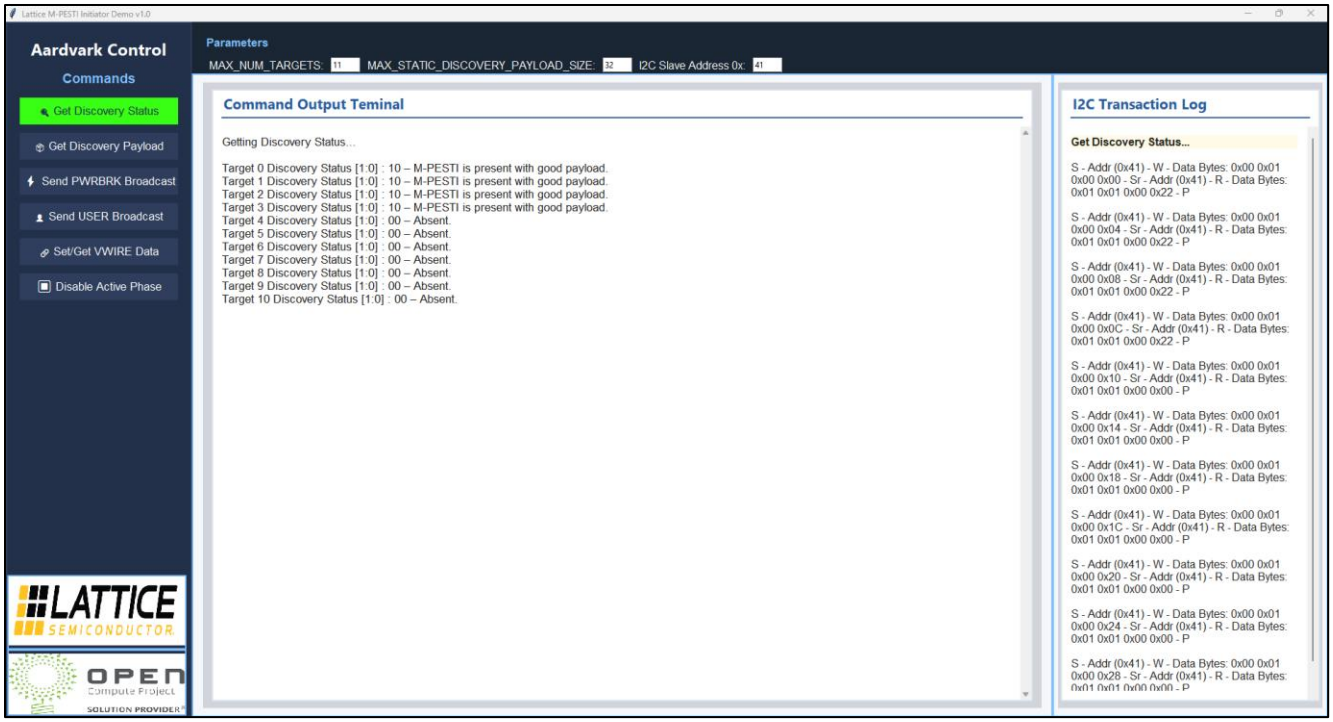

Figure 6.10. Good Payload Discovery Status

6.2.2. Get Discovery Payload

As shown in the following figure, when the discovery payload has been successfully received during the Get Discovery Status, a summary is displayed on the screen, including the Payload Size and the actual Payload Read. Note that the reference code continues to follow the Payload Format of Specification 1.0, because Specification 1.2 does not yet fully define the information about the number of Virtual Wires supported.

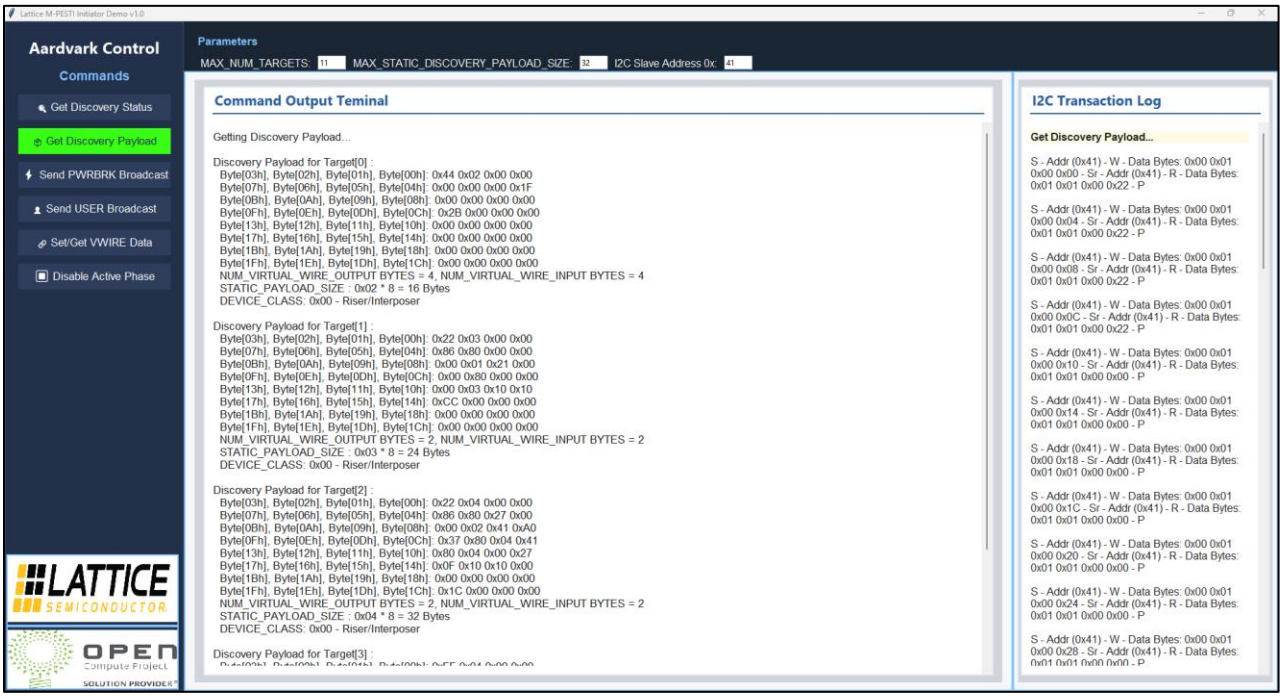

Figure 6.11. Successful Get Discovery Payload

6.2.3. Send PWRBRK Broadcast

This menu selection allows the broadcasting of PWRBRK Command (0x00) to all targets. The command frame is preceded by the BROADCAST Command (0xFF), followed by the PWRBRK Command (0x00).

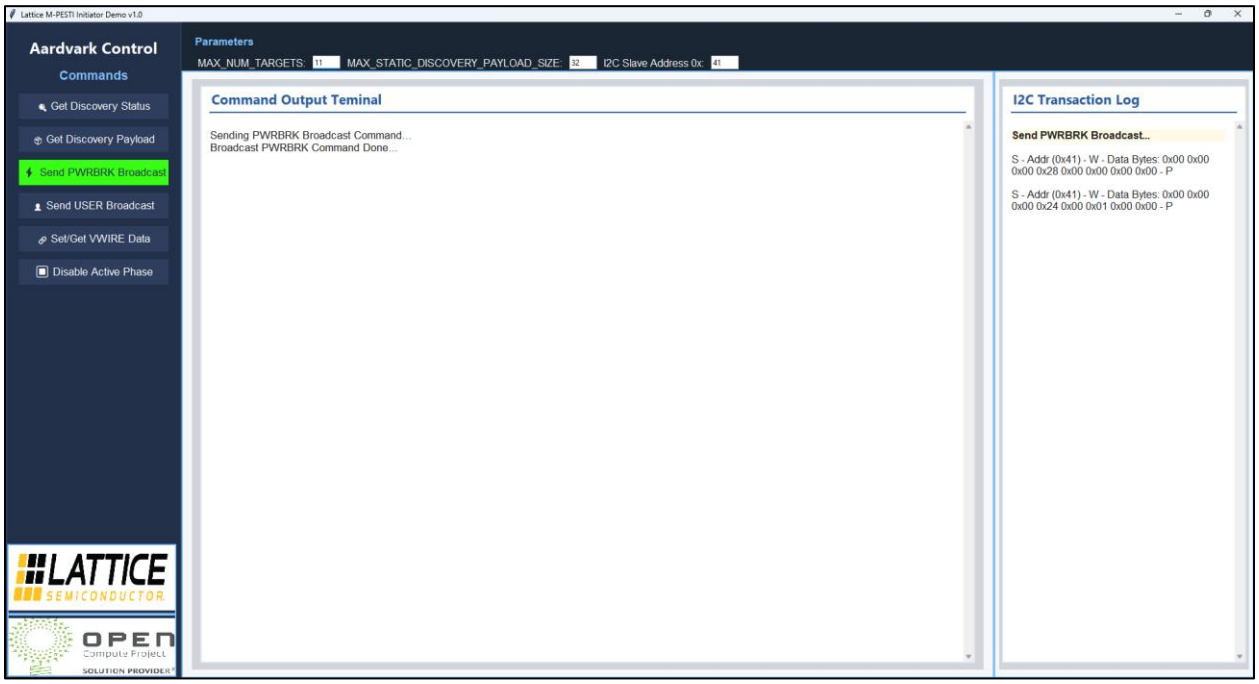

Figure 6.12. Successful PWRBRK Broadcast Command

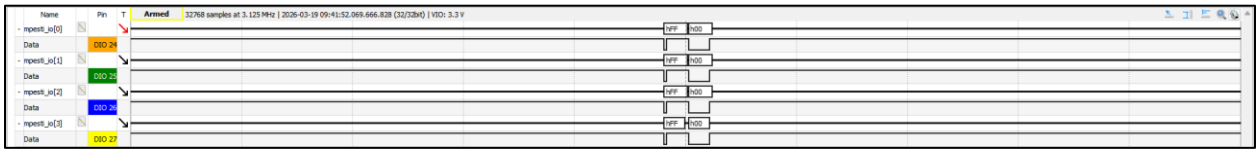

Figure 6.13. Successful PWRBRK Broadcast Hardware Waveform

6.2.4. Send USER Broadcast

This menu selection allows the broadcasting of USER Command to all targets. The command frame is preceded by the BROADCAST Command (0xFF), followed by the USER Command (User input).

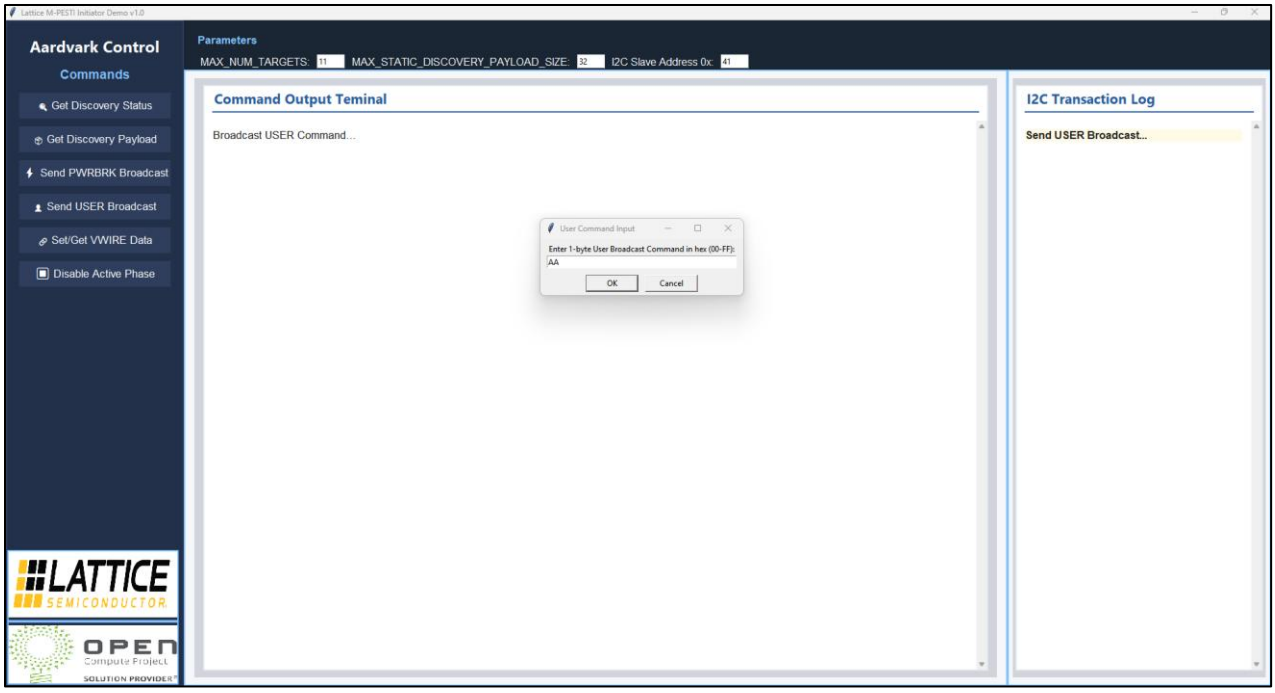

Figure 6.14. Sending USER Broadcast User Input

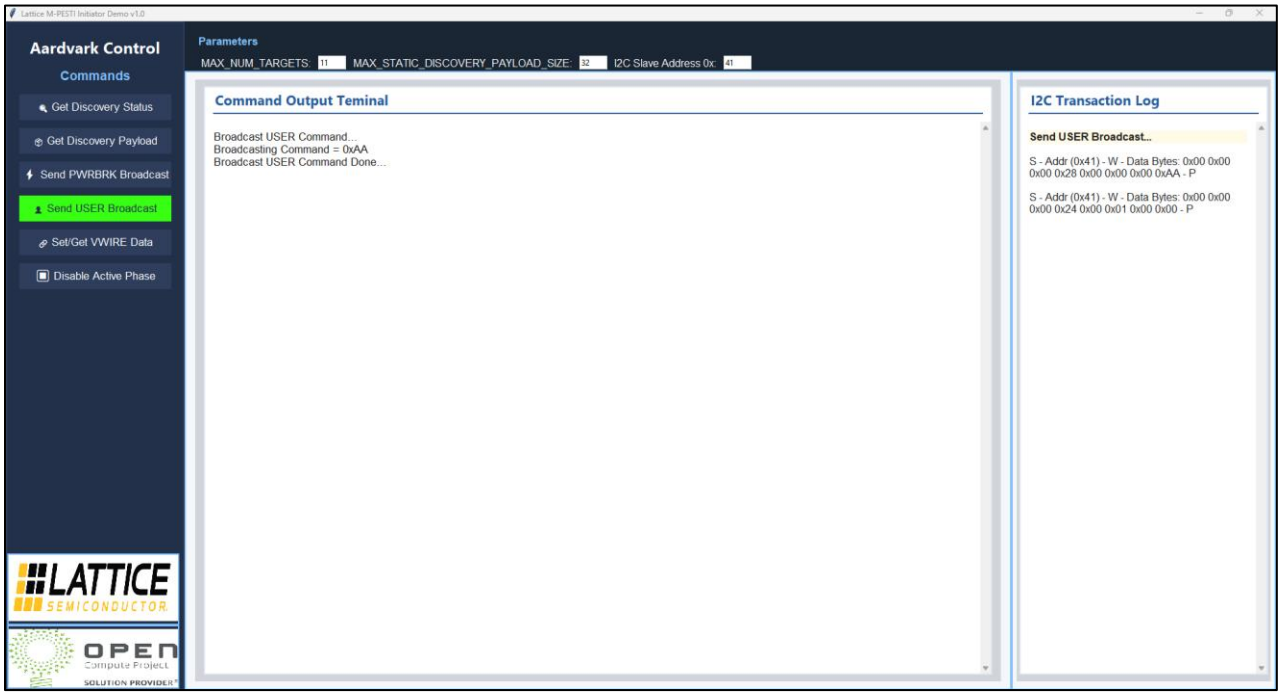

Figure 6.15. Successful USER Broadcast

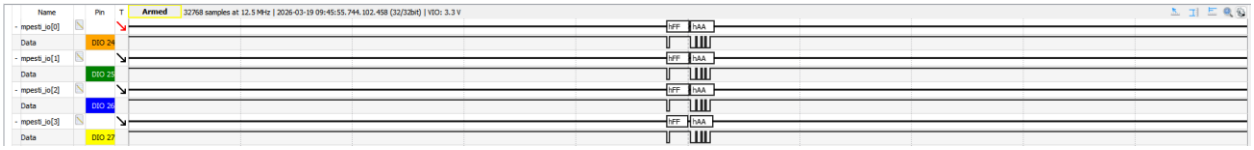

Figure 6.16. Successful USER Broadcast Hardware Waveform

6.2.5. Set/Get VWIRE Data

This menu selection allows the MPESTI Initiator to issue the Active Phase command (0x01) and exchange Virtual Wire data with multiple targets in a round-robin fashion. The reference code automatically parses the supported number of Virtual Wires for each target by reading the byte 03h of the target's Discovery Payload, specifically, the NUM_VIRTUAL_WIRE_OUTPUT_BYTES[3:0] and NUM_VIRTUAL_WIRE_INPUT_BYTES[3:0].

The MPESTI Initiator uses this information to determine the number of the Virtual Wire bytes to send and the number of the Virtual Wire bytes to receive. This menu selection also provides the option to specify the Virtual Wire data values to be sent, as shown in the following figures.

Note that the reference code continues to follow the Payload Format of Specification 1.0, because Specification 1.2 does not yet fully define the information about the number of Virtual Wires supported.

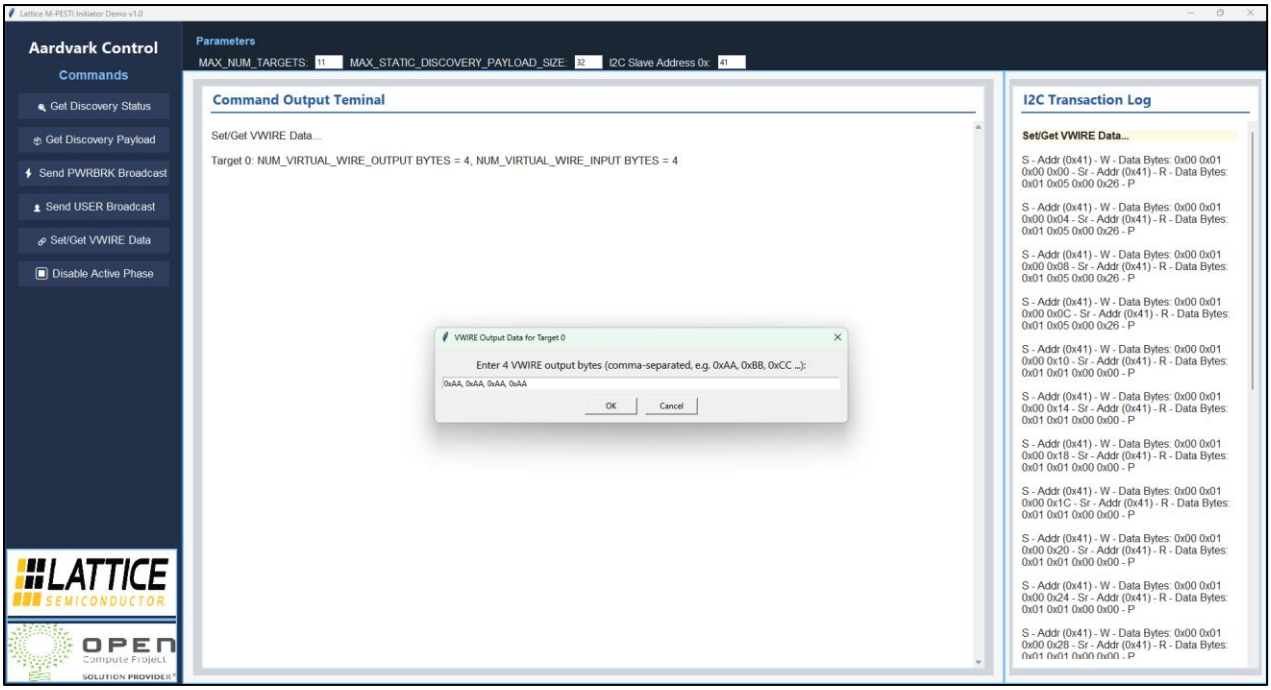

Figure 6.17. Set/Get Virtual Wire User Input

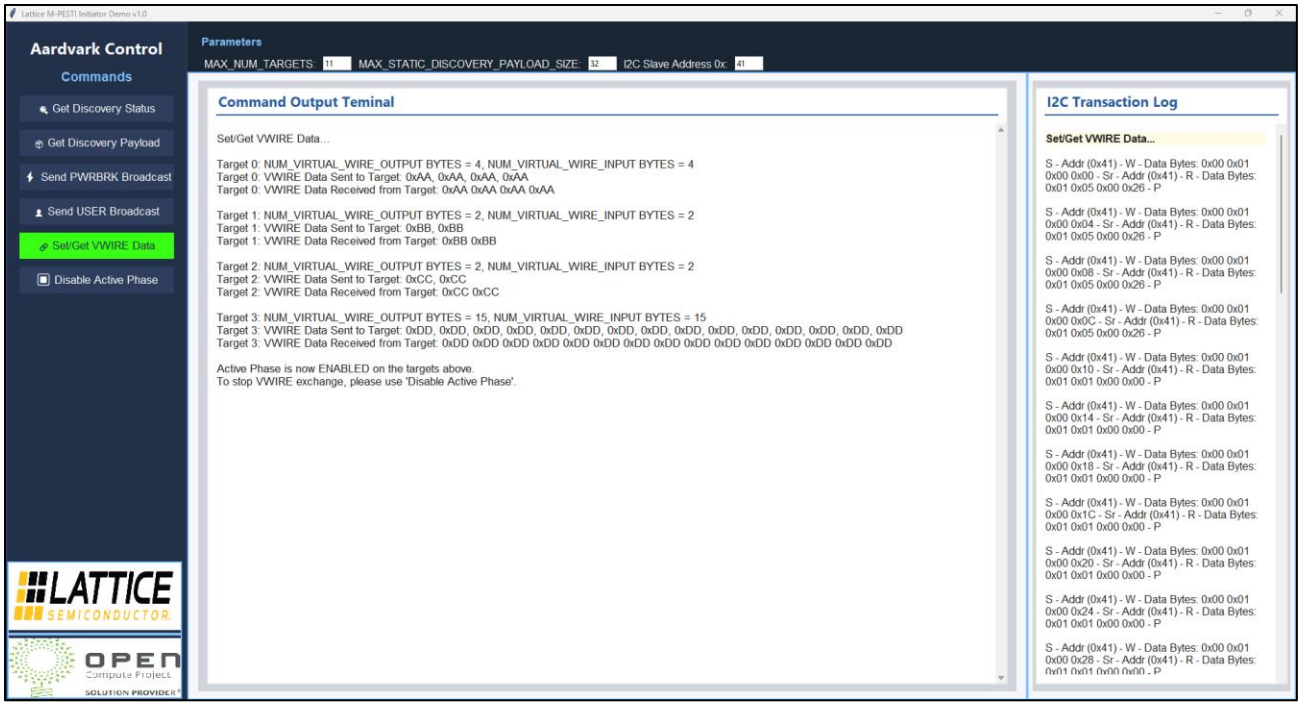

Figure 6.18. Successful Set/Get Virtual Wire

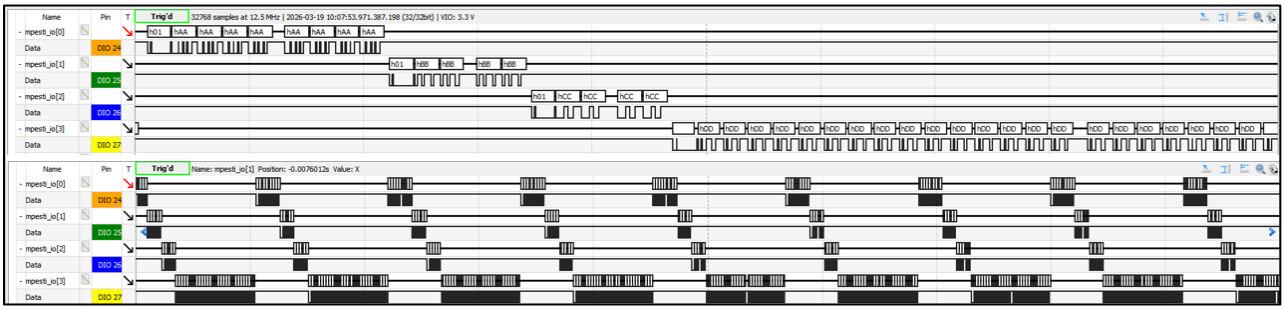

Figure 6.19. Set/Get Virtual Wire Hardware Waveform

Note: Once this selection is successfully completed, the Active Phase bit of the displayed targets is enabled, which means the Virtual Wire Exchange is performed continuously until it is disabled. For more information, refer to the Disable Active Phase section.

6.2.6. Disable Active Phase

This menu selection allows you to stop the Virtual Wire Exchange by setting the Active Phase Enable bit to 0 for each of the target currently in Active Phase (continuous Virtual Wire Exchange). The continuous Virtual Wire Exchange (shown in Figure 6.19) stops, once this operation completes successfully, as shown in Figure 6.21.

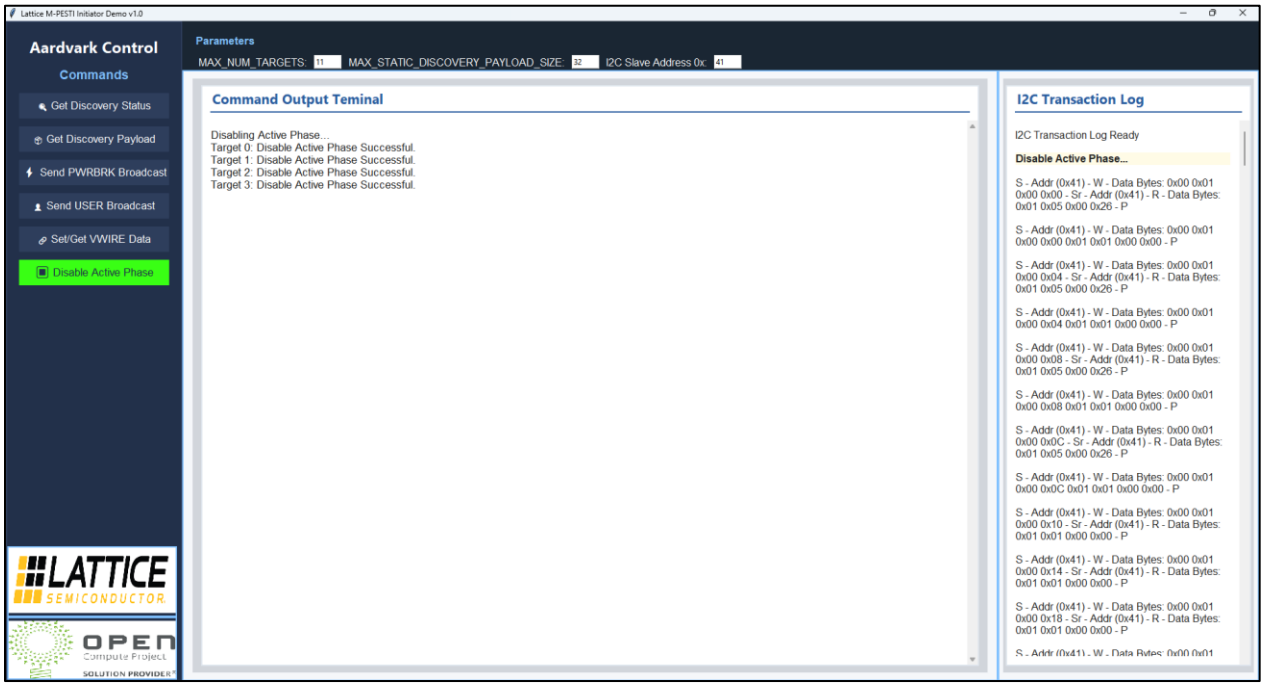

Figure 6.20. Disable Active Phase Successful

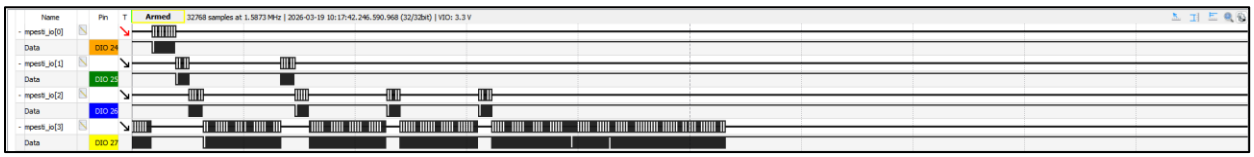

Figure 6.21. Disable Active Phase Hardware Waveform

7. I2C Transactions

This design utilizes the I2C to APB Bridge reference design that converts I2C transactions into APB requestor transactions to gain access and control to the M-PESTI IP core register address map. The I2C timing diagrams and example register access are shown in the following figures.

7.1. Write Operation

The figure below shows the I2C framing for a single APB write transaction. The figure is described as the following:

- The first four bytes of serial data are converted into a 32-bit register address (or APB Address)
- The next four bytes of serial data are converted into 32-bit write data (APB Write Data) to that register address (or APB Address)
- For multiple write operations, this framing must be repeated by the I2C Controller for all APB address locations

For more information, refer to the I2C to APB Bridge Reference Design web page.

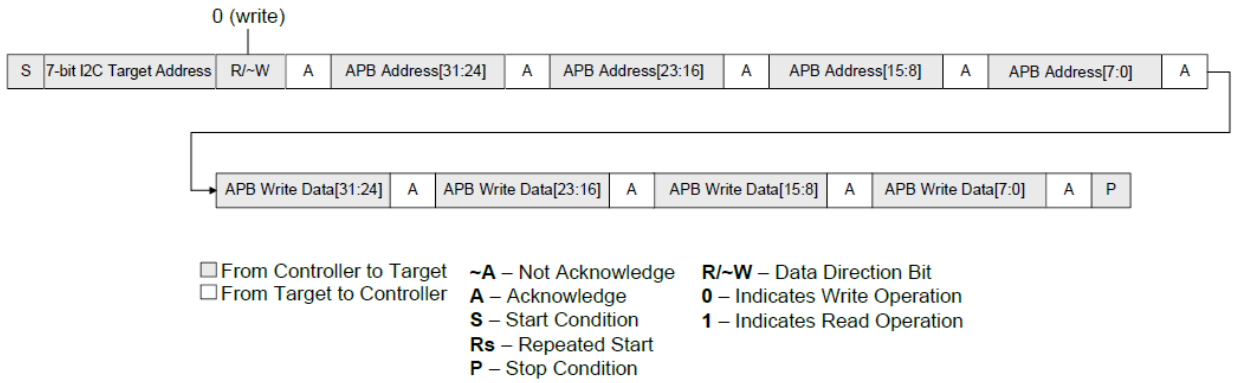

Figure 7.1. Write Operation Data Format for I2C to APB Conversion

7.2. Read Operation

The figure below shows the I2C framing for a single APB read transaction. The figure is described as the following:

- The first four bytes of I2C serial data are converted into the 32-bit register address (or APB Address)
- The next four bytes of I2C serial data contain the 32-bit read data (APB Read Data) coming from the accessed register address (or APB Address)
- For multiple read operations, this framing must be repeated by the I2C Controller for all APB address locations.

For more information, refer to the I2C to APB Bridge Reference Design web page.

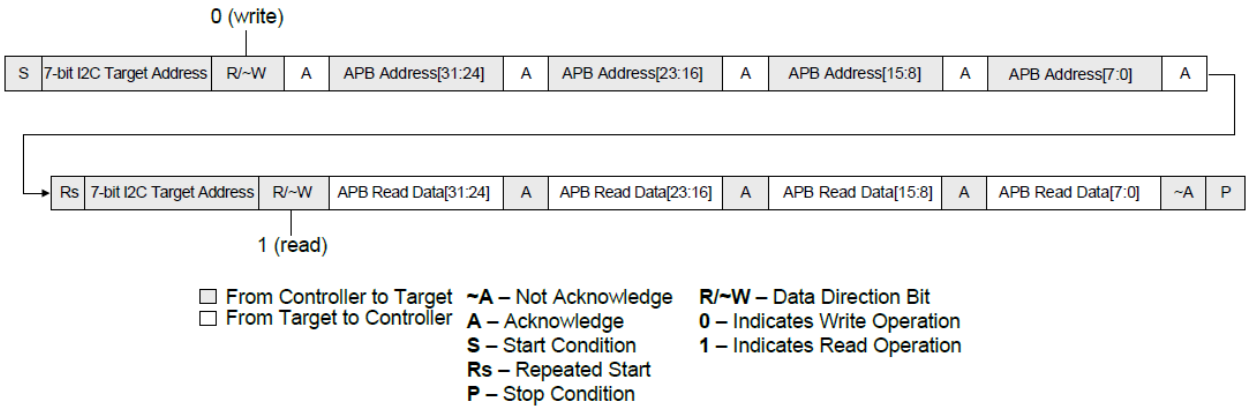

Figure 7.2. Read Operation Data Format for I2C to APB Conversion

7.3. Register Write Operation Example

The figure below shows an example I2C transaction that performs a write operation to the Required Registers of M-PESTI Initiator Target 0. For more information about the M-PESTI Initiator register descriptions, refer to the M-PESTI Initiator IP User Guide (FPGA-IPUG-02258).

Register Address: 0x0001_0000 (Target 0 Required Registers)

Write Data: 0x0101_0000

- Discovery Phase Enable [16] = 1
- Active Phase Enable [18] = 0
- Broadcast Subscription [24] = 1

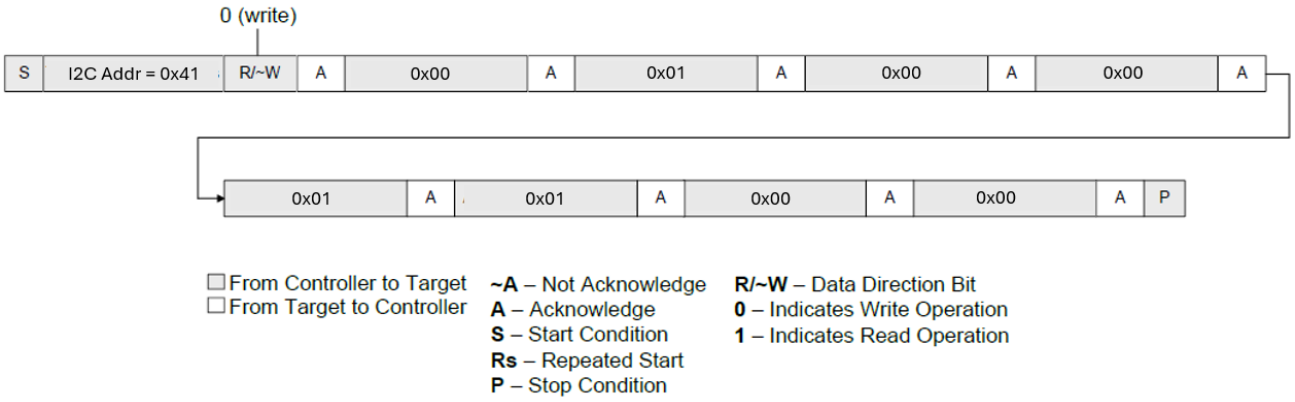

Figure 7.3. Write Data 0x0101_0000 to Target 0 Required Registers (0x0001_0000)

7.4. Register Read Operation Example

The figure below shows an example I2C transaction that performs a read operation to the Required Registers of M-PESTI Initiator Target 0. For more information about the M-PESTI Initiator register descriptions, refer to the M-PESTI Initiator IP User Guide (FPGA-IPUG-02258).

Register Address: 0x0001_0000 (Target 0 Required Registers)

Read Data: 0x0105_0026

- Discovery Status [1:0] = 2'b10
- Active Phase Status [2] = 1
- Presence Timeout Detection [5] = 1
- Discovery Phase Enable [16] = 1
- Active Phase Enable [18] = 1
- Broadcast subscription [24] = 1

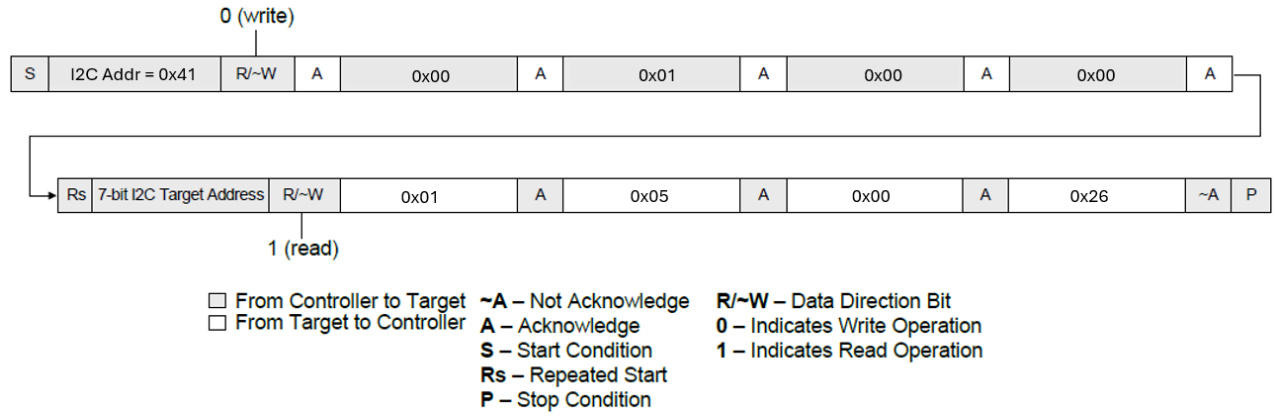

Figure 7.4. Read Target 0 Required Registers (0x0001_0000)

8. Simulating the Reference Design

This reference design includes pre-generated RTL files that can be directly simulated using the ModelSim™ Lattice FPGA Edition software or QuestaSim™ Lattice FPGA Edition software. To simulate the design, perform the following steps:

1. Unzip the reference design zip file.
2. Open the *.do* file in a text editor and replace the *SET THE SIMULATION PATH HERE* text in line 2 with the directory path of the simulation file. Figure below shows an example of the replaced directory path in line 5.

```
2  set SIM_DIR "SET THE SIMULATION PATH HERE"
3
4  # Example:
5  # set SIM_DIR "C:/Users/M-PESTI_Initiator_Reference_Design/Simulation"
```

Figure 8.1. Changing the Simulation Directory

3. Launch the ModelSim Lattice FPGA Edition software and select **Tools > TCL > Execute Macro**.

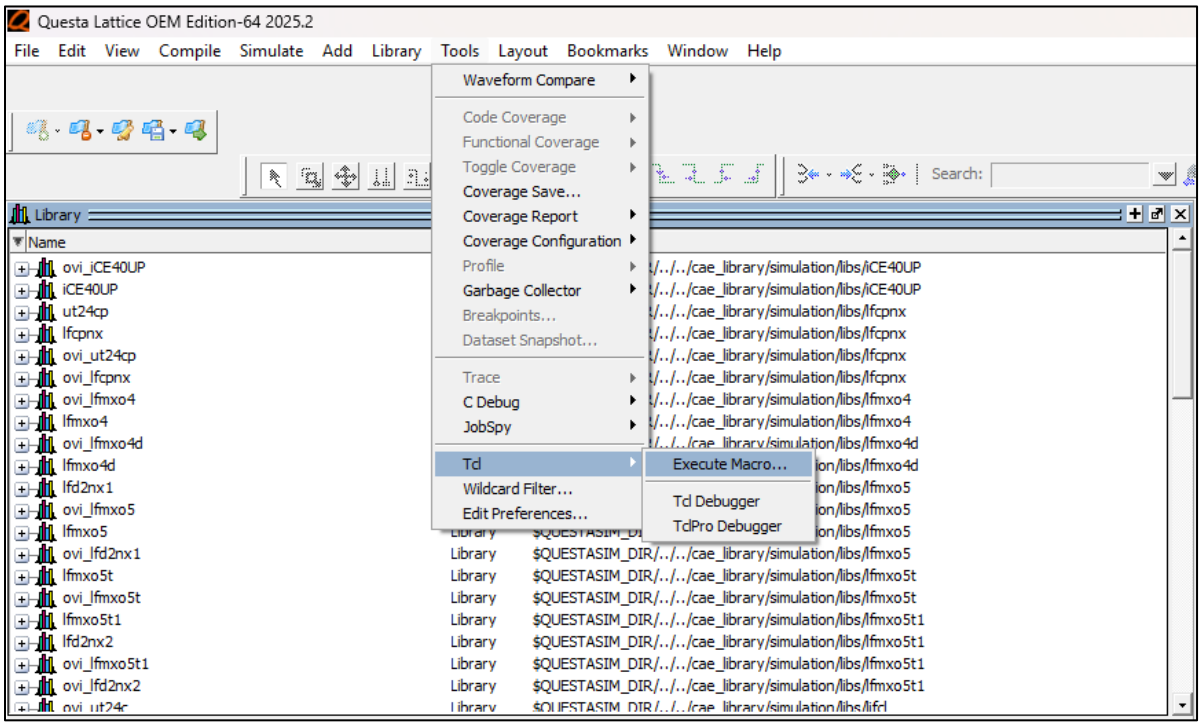

Figure 8.2. Running the Simulation Script File

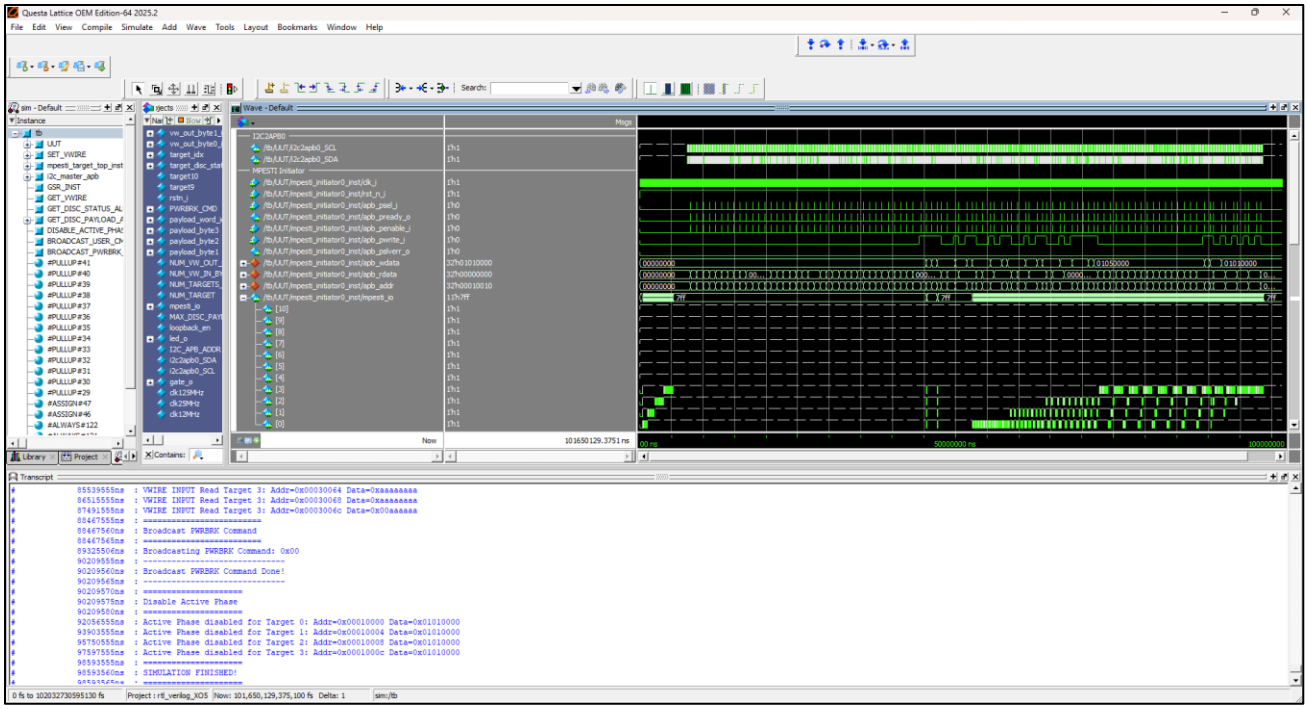

Figure 8.3. Simulation Waveform

9. Resource Utilization

The following table lists the resource utilization information of the reference design.

Table 9.1. Resource Utilization

Module or IP	MachXO3LF/ MachXO3D Utilization		MachXO5-NX Utilization		Notes
	LUTs	EBR	LUTs	EBR	
M-PESTI IP	2311	9	2605	4	The current design supports up to 11 targets. This limit can be increased through the IP GUI.
I2C to APB Bridge Reference Design	198	—	207	—	The current 7-bit Slave Address is set to 0x41. This value can be changed as needed.
Led Counter	22	—	58	—	Used as an indicator that the clock is present, the PLL has achieved a locked state, and that the bitstream is running.
PLL	2	—	2	—	• In MachXO3LF and MachXO3D devices, the PLL is used to generate the 25 MHz system clock from the external (onboard) 12 MHz oscillator • In MachXO5-NX devices, the PLL is used to generate the 25 MHz system clock from the external (onboard) 125 MHz oscillator

References

- M-PESTI Initiator IP User Guide (FPGA-IPUG-02258)
- M-PESTI Initiator IP Release Notes (FPGA-RN-02005)
- M-PESTI Initiator Driver API Reference (FPGA-TN-02413)
- I2C to APB Bridge Reference Design web page
- M-PESTI Initiator IP web page
- Open Compute Project® web page
- MachXO3 Family Devices (FPGA-DS-02026)
- MachXO3D Family Devices (FPGA-DS-02032)
- MachXO3LF Starter Kit (FPGA-EB-02036)
- MachXO3D Breakout Board (FPGA-UG-02084)
- MachXO5-NX Development Board (FPGA-EB-02052)
- MachXO3D web page
- MachXO3 web page
- MachXO5-NX web page
- Lattice Radiant FPGA design software
- Lattice Solutions Reference Designs web page
- Lattice Solutions IP Cores web page
- Lattice Propel Design Environment web page
- Lattice Propel 2025.2 Builder User Guide (FPGA-UG-022434)
- Lattice Radiant Software User Guide

Technical Support Assistance

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

For frequently asked questions, please refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.0, April 2026

Section	Change Summary
All	Initial release.

www.latticesemi.com