Using TraceID

Technical Note

FPGA-TN-02084-2.5

January 2023

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 associated risk the responsibility entirely of the 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
EFB	Embedded Function Block
I^2C	Inter-integrated circuit
IP	Intellectual property
JTAG	Joint Test Action Group
LMMI	Lattice Memory Mapped Interface
OEM	Original equipment manufacturer
PLD	Programmable Logic Device
SPI	Serial Peripheral Interface
SSPI	Security Support Provider Interface

1. Introduction

Design theft has caused many companies to explore methods to insure that their designs and intellectual property (IP) are protected or less prone to blatant copying.

Design theft occurs when a design is copied in part or as a whole and then designed into a cheaper competing product.

The MachXO2™, MachXO3™, ECP5™, ECP5-5G™, CrossLink™-NX, Certus™-NX, CertusPro™-NX, MachXO5™-NX, and Avant™ FPGA families have a feature called TraceID for securing original design and IP. TraceID is a unique, 64-bit code that is programmed during the manufacturing of the device, thus linking a specific design to a specific device. This ensures that only the original product manufacturer who ordered a specific device has access to it.

2. Why is TraceID Important?

TraceID can be used to prevent overbuilding and cloning of user designs. Overbuilding occurs when a contract manufacturer builds more products than the original company has approved. These extra products are, in turn, sold through other channels for profit without the knowledge or consent of the original company. Cloning is the act of making exact copies of a product and selling them under a different name at a lower price (thus reducing the OEM profit). These practices cause OEMs to lose money not only from lost sales and lower margins, but also from unseen support costs such as failure analysis.

The TraceID feature can be used to prevent both overbuilding and cloning.

3. How Does TraceID Work?

TraceID is a unique, 64-bit device identification tracking number which is stored in the feature row of the device.

The 64 bits contain the following unique information:

- Wafer lot number
- Wafer number within the lot
- Die X location
- Die Y location
- User-defined design-specific code

The TraceID register format is shown in Table 3.1.

Table 3.1. TraceID Register

[63:56]	8 bits, User-Defined Code
[55:24]	32 bits, Wafer Lot Number
[23:19]	5 bits, Wafer Number (within the lot)
[18:12]	7 bits, Wafer Die X Location
[11:5]	7 bits, Wafer Die Y Location
[4:0]	5 bits, Extra Spare Bits

The most significant eight bits are the user-defined design-specific code. These eight bits are read and write accessible. The remaining 56 bits are read-only and are programmed at the time the device is manufactured by Lattice. The 8-bit user-defined code plus the 56-bit factory-programmed portion together guarantee that every device has a unique TraceID. This uniqueness provides OEMs greater control over how many of their products are introduced in the market and the ability to detect false products.

4. How to Program User-Defined Code of the TraceID for MachXO2

Lattice design software can be used to set a specific 8-bit user-defined code in the TraceID register. The TRACE_ID_BINARY preference must be set to a chosen value in the LPF file.

In the LPF file, set the TraceID value using the following format:

```
TRACEID "<8-bit value>"
```

Below is an example:

```
TRACEID "01101001"
```

When a programming file is created, the 8-bit TraceID value is embedded in the JED file feature row. During programming,

the TraceID registers in the device are updated with the one in the JED. The default value for the user defined code is " 00000000".

5. Accessing the TraceID Register

The TraceID value in the MachXO2 and MachXO3 devices can be read using the internal WISHBONE port or externally through the JTAG, SSPI or I^2C ports. The TraceID value in ECP5 and ECP5-5G devices can be read through JTAG or SSPI port since ECP5 and ECP5-5G does not have I^2C port nor internal WISHBONE. The TraceID value in CrossLink-NX, Certus-NX, CertusPro-NX, and MachXO5-NX devices can be read using the JTAG, SSPI, or I^2C ports. The TraceID value in Avant can be read using JTAG, SSPI, or Lattice Memory Mapped Interface (LMMI) ports.

For SSPI, I^2C, and JTAG interfaces, the UIDCODE_PUB command must be used to read the TraceID register. The OPCODE for the UIDCODE_PUB command is "00011001".

The TraceID value can also be read using either the Lattice Diamond® Programmer tool or the Lattice Radiant® Programmer tool, depending on the device family. Figure 5.1 and Figure 5.2 show the settings required to read the TraceID through the Diamond Programmer and Lattice Radiant Programmer tools, respectively.

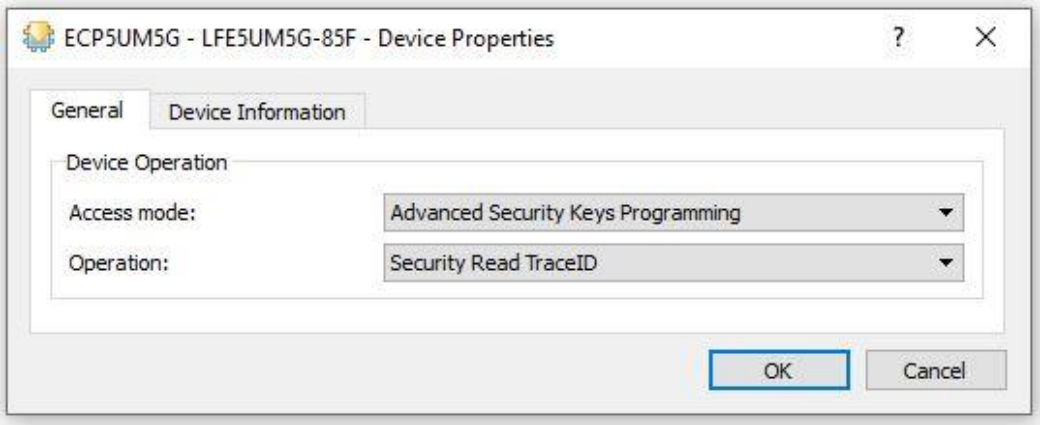

Figure 5.1. Read TraceID through Diamond Programming Tool

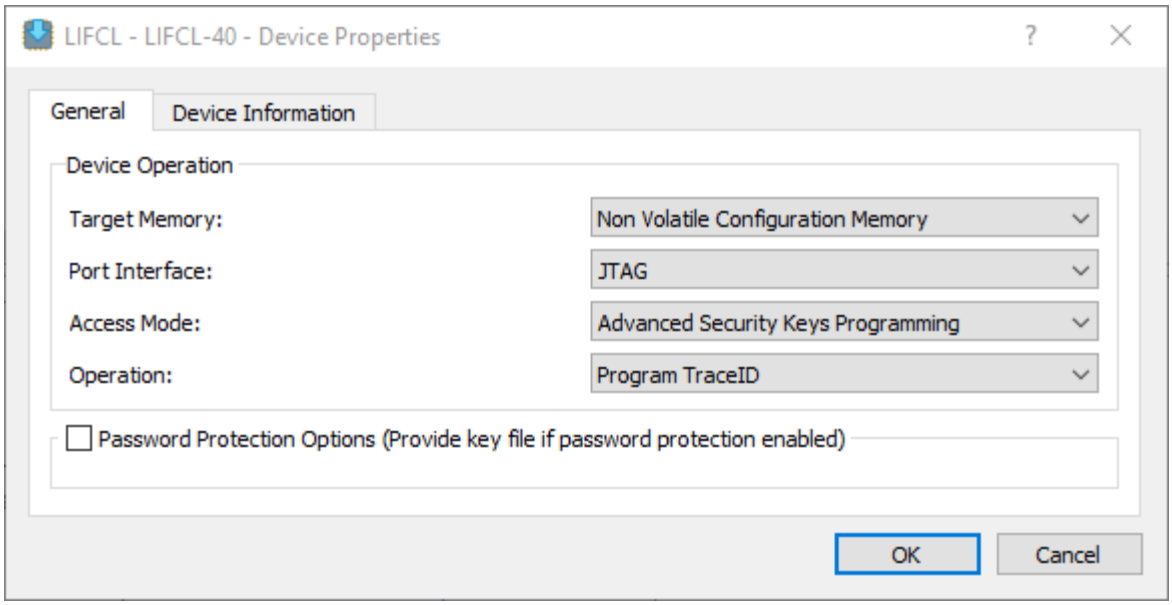

Figure 5.2. Program TraceID through Lattice Radiant Programmer Tool

For MachXO2, MachXO3, CrossLink-NX, Certus-NX, CertusPro-NX, and MachXO5-NX bits [63:56] of the TraceID can be written through the Feature Row. In Diamond Programmer, select **Advanced Security Keys Programming** in **Access Mode** and **Security Program Feature Rows** in **Operation**. In Radiant Programmer, select **Advanced Security Keys Programming** in **Access Mode** and **Program TraceID** in **Operation**. When the operation is run, a window similar to Figure 5.3 opens. For Avant, bits [63:56] of the TraceID can be programmed through Radiant Programmer by selecting **Feature Rows Programming** in **Access Mode** and **Program Feature Row** in **Operation** (Figure 5.4).

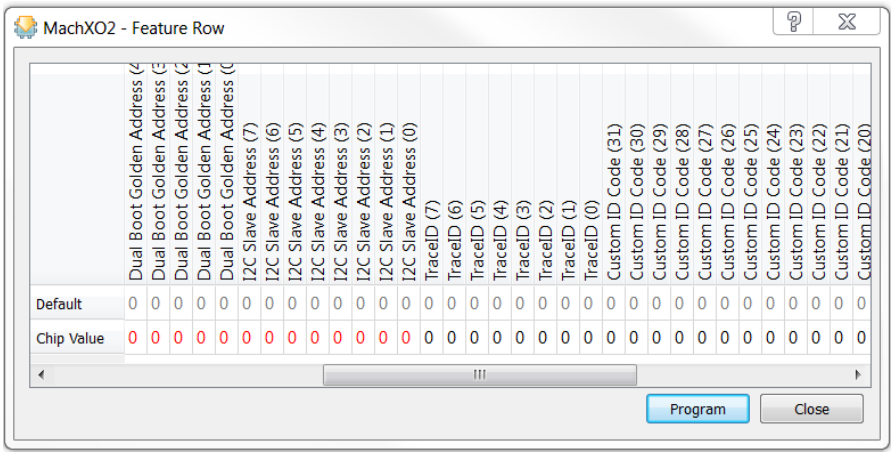

Figure 5.3. Feature Row Editor

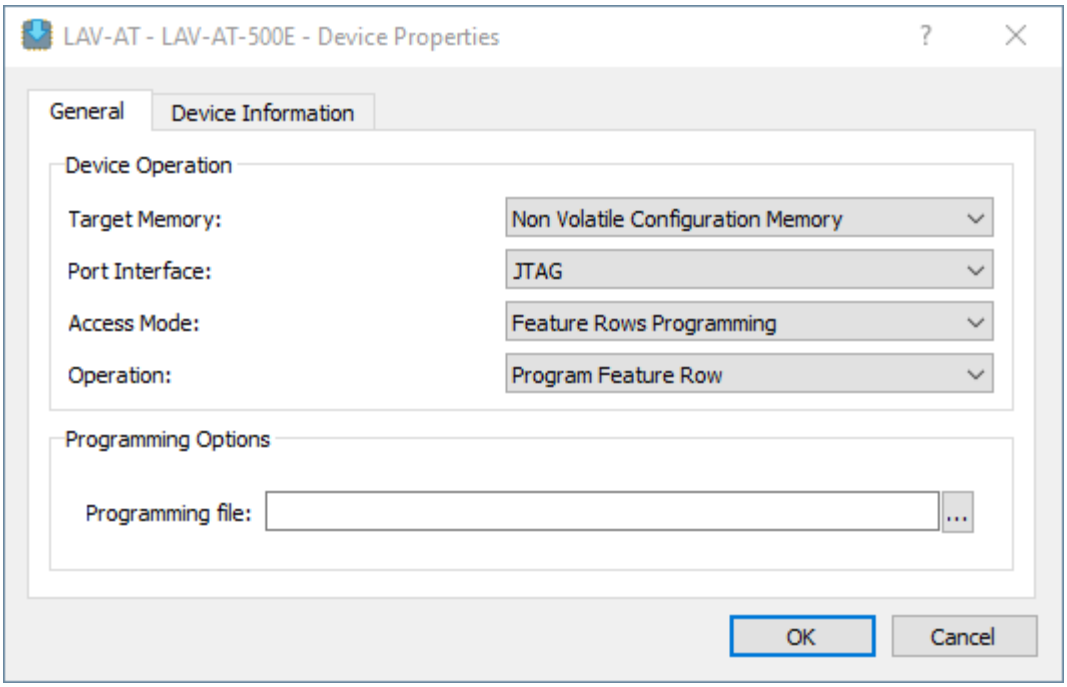

Figure 5.4. Avant - Program Feature Row through Lattice Radiant Programmer Tool

5.1. TraceID Access Through the JTAG Port

The JTAG port has access to all configuration logic resources including the TraceID. To read the TraceID using JTAG, shift the command (0x19h) into the instruction register then read the 64-bit TraceID out of the data register.

5.2. TraceID Access Through the WISHBONE Slave Interface (MachXO2 and MachXO3 Only)

The WISHBONE Slave interface of the EFB module enables designers to access the TraceID directly from the PLD core logic. The WISHBONE bus signals are utilized by a WISHBONE host that designers can implement using the general purpose PLD resources. In addition to the WISHBONE bus signals, an interrupt request output signal is brought to the PLD fabric.

The WISHBONE interface communicates to the configuration logic through a set of data, control and status registers. Table 5.1 shows the register names and their functions. These registers are the subset of the EFB register map. The details of the WISHBONE slave interface pins, EFB register map, and WISHBONE register definition can be found in Using User Flash Memory and Hardened Control Functions in MachXO2 Devices (FPGA-TN-02162) and Using Hardened Control Functions in MachXO3 Devices (FPGA-TN-02063).

Table 5.1. WISHBONE Register

WISHBONE to CFG Register Name	Register Function	Address	Access
CFGCR	Control	0x70	Read/Write
CFGTXDR	Transmit Data	0x71	Write
CFGSR	Status	0x72	Read
CFGRXDR	Receive Data	0x73	Read
CFGIRQ	Interrupt Request	0x74	Read/Write
CFGIRQEN	Interrupt Request Enable	0x75	Read/Write

When using the WISHBONE bus interface, the opcodes, operand, and data are written to the CFGTXDR register. This is required only when communicating with the configuration logic inside the MachXO2 and MachXO3 devices. The TraceID can be accessed through the WISHBONE interface by writing the opcode and operand into the CFGTXDR register. The TraceID information can then be read from the CFGRXDR register.

The opcode to access the TraceID is 0x19h and the operand is 0x000000h.

5.3. TraceID Access Through the Slave SPI Port

The Slave SPI port can be used to perform read operations to the TraceID. The configuration SPI port is shared with the hardened SPI core of the EFB module. Asserting the configuration SN (select) pin causes the SPI port to transition its service from user mode to configuration mode. The TraceID can be accessed using the SPI port by following the command sequence described below.

To access the TraceID:

1. Pull down the configuration SN select pin (SPI Slave Select)
2. Send 8'h19 command from the external SPI master
3. Send 24-bit operand 24'h000000
4. Receive TraceID from SPI slave in next 64 SCLK cycle
5. Pull up configuration SN select pin

The complete sequence is shown in Figure 5.5 and Figure 5.6.

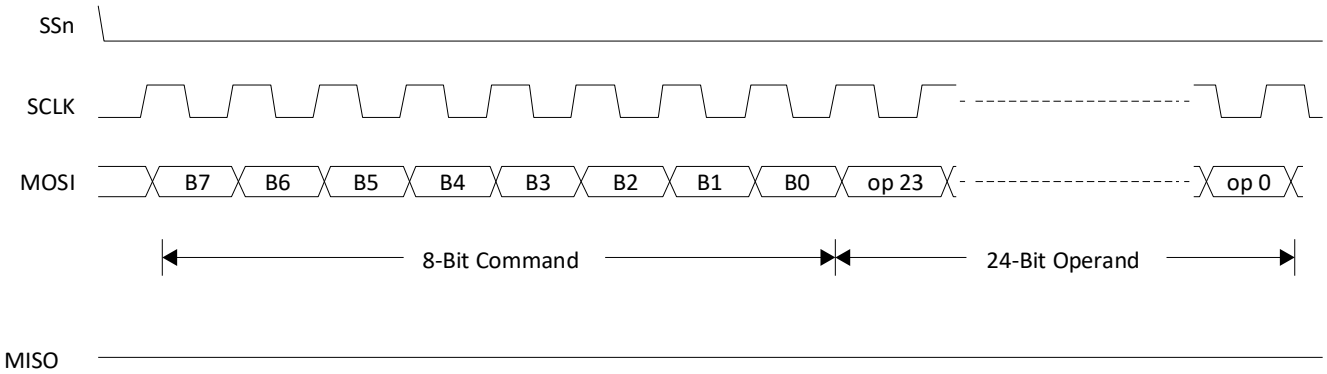

Figure 5.5. TraceID Read through SPI

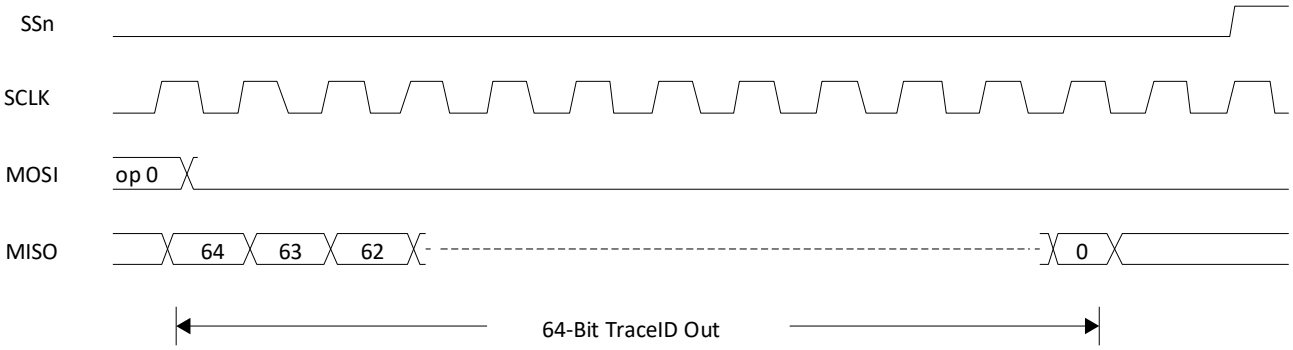

Figure 5.6. TraceID Read through SPI, Continued

5.4. TraceID Access Through the I²C Port (MachXO2, MachXO3, CrossLink-NX, Certus-NX, CertusPro-NX, and MachXO5-NX Only)

All MachXO2, MachXO3, CrossLink-NX, Certus-NX, CertusPro-NX, and MachXO5-NX devices have an I²C port, which can be used to perform read operations to the TraceID.

For MachXO2 and MachXO3, the configuration I²C port is shared with the hardened I²C primary core of the EFB module. Addressing the I²C primary port with the configuration address changes the port service from user mode with the WISHBONE interface to configuration mode. The pin locations of the configuration I²C port are pre-assigned in all MachXO2 and MachXO3 devices.

For CrossLink-NX, Certus-NX CertusPro-NX, and MachXO5-NX devices, the configuration I²C port is separate from the hard I²C block accessible from user logic, and has pre-defined pins. Sending the I²C address followed by the correct preamble activates the port. The I²C configuration port is a shared interface to the configuration logic with the configuration SPI port, and only one can be active at a time.

There is one address byte required since 7-bit addresses are used. The last bit of the address byte is the read/write bit and should always be set according to the required operation. This 7-bit I²C address is 1000000 (80h) which is the default address. The read sequence uses a repeated start condition during the sequence to avoid bus release during communication. For 10-bit addressing the I²C slave address is 10'b1111000000.

To read the TraceID through the I²C bus:

1. Send start condition.
2. Send default slave address (8'h80) and write command.
3. Send the 8-bit command 8'h19.
4. Send the 24-bit operand 24'h000000 in three single-byte transfers.
5. Send repeated start.
6. Send the slave address and read command.
7. Read the first through seventh bytes of the TraceID and send ack for each byte read.
8. Read the last TraceID byte and send nack.
9. Send the stop command.

Figure 5.7 shows the TraceID read through I²C.

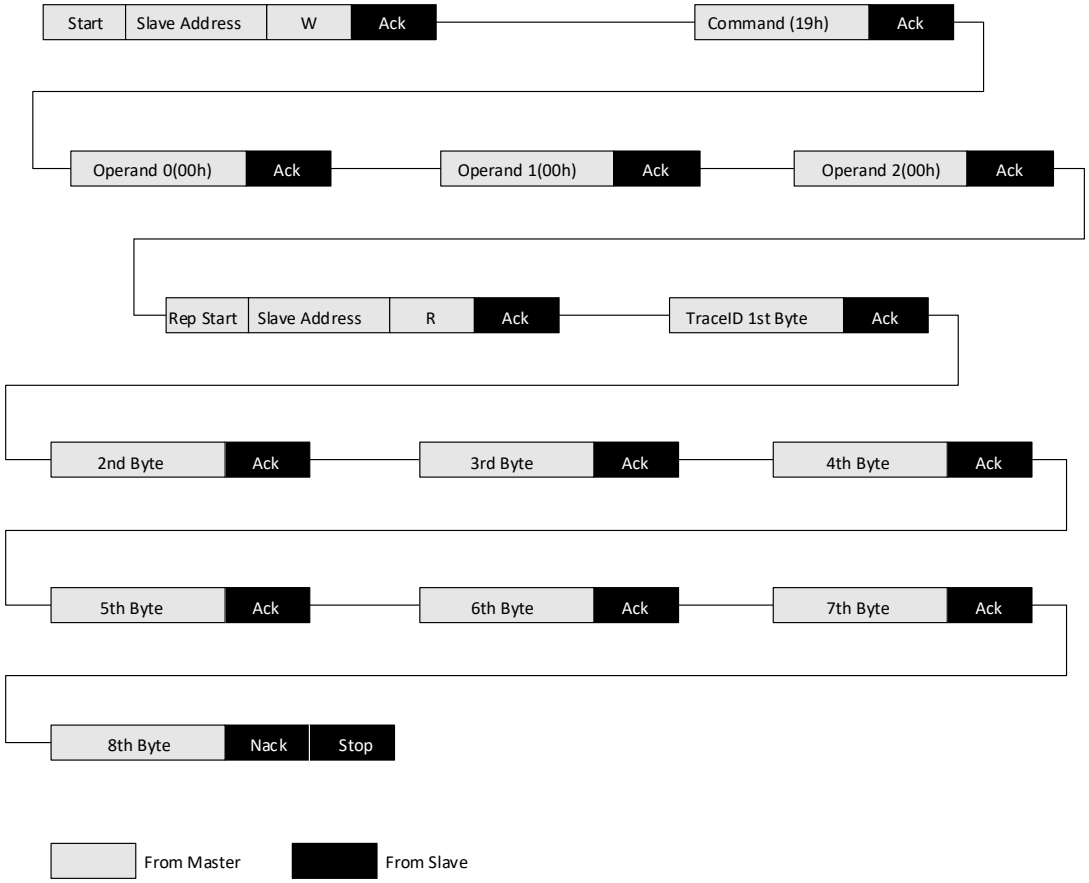

Figure 5.7. TraceID Read through I²C

5.5. TraceID Access Through the LMMI (Avant Only)

The Lattice Memory Mapped Interface (LMMI) is a memory-mapped address/data interface. It defines a standard set of interface signals for register/memory access and supports both single and burst transactions. LMMI allows the designer to access the TraceID directly from the PLD core logic. The opcode to access the TraceID is 0x19h. Details on LMMI can be found in Lattice Memory Mapped Interface and Lattice Interrupt Interface User Guide (FPGA-UG-02039).

6. MachXO2 Example Uses of TraceID

TraceID can be used to validate that the MachXO2 device or the system in general, as authorized by the OEM.

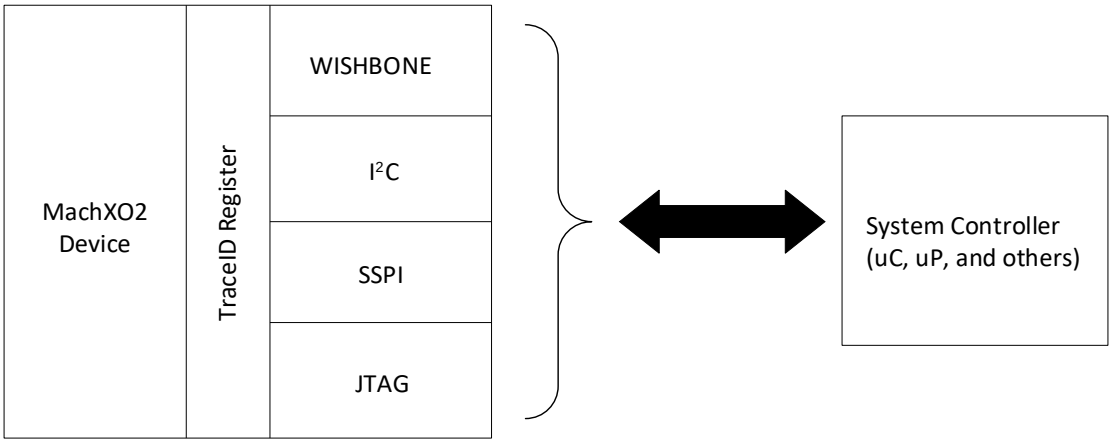

Figure 6.1. TraceID Read through I²C

One way of implementing this is to use the system controller, either a microcontroller or microprocessor, to access the TraceID register (through WISHBONE, I²C, SPI or JTAG interfaces) and compare it against a list of approved TraceID device tables. If the TraceID matches the approved device list, the system can continue to function as intended.

In cases where the read TraceID does not match with the approved device list, the system controller can choose to log the event and take one of the following actions:

- Stall – Stop working
- Continue with limited functionality – Partial operation of system
- Erase or destroy integral data in the system – Erase the boot ROM, Flash memory, register tables, etc.

Technical Support Assistance

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

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

Revision History

Revision 2.5, January 2023

Section	Change Summary
Accessing the TraceID Register	Fixed the figure reference error for Figure 5.1. Read TraceID through Diamond Programming Tool and Figure 5.2. Program TraceID through Lattice Radiant Programmer Tool.
Technical Support Assistance	Removed https: from the Technical Support Assistance Section.

Revision 2.4, November 2022

Section	Change Summary
All	Updated formatting styles.
Accessing the TraceID Register	Added link to FPGA-UG-02039 in Section 5.5 TraceID Access Through the LMMI (Avant Only).

Revision 2.3, May 2022

Section	Change Summary
All	Added MachXO5-NX device family.
Accessing the TraceID Register	Updated the document numbers of referenced technical notes.

Revision 2.2, June 2021

Section	Change Summary
All	Added CertusPro-NX device family.

Revision 2.1, October 2020

Section	Change Summary
Accessing the TraceID Register	• Updated process of writing TraceID through Feature Row. • Updated Figure 5.2. Read TraceID through Lattice Radiant Programmer Tool. • Corrected reference to Figure 5.6. TraceID Read through I2C.

Revision 2.0, June 2020

Section	Change Summary
All	• Added Certus-NX device family. • Minor changes in formatting and style.

Revision 1.9, December 2019

Section	Change Summary
All	• Changed document number from TN1207 to FPGA-TN-02084. • Updated document template. • Updated document links.
Disclaimers	Added this section.
Introduction	Added CrossLink-NX device family.
Accessing the TraceID Register	• Added CrossLink-NX device family. • Added Figure 5.2. Read TraceID through Lattice Radiant Programmer Tool.

Revision 1.8, October 2015

Section	Change Summary
Introduction	Added ECP5-5G device family.
Accessing the TraceID Register	Added ECP5-5G device family.
Technical Support Assistance	Updated this section.

Revision 1.7, March 2015

Section	Change Summary
All	Product name adjustment. Included MachXO3LF device.

Revision 1.6, June 2014

Section	Change Summary
All	Corrected typographical errors on product name.

Revision 1.5, April 2014

Section	Change Summary
All	• Changed document title to Using TraceID. • Added support for MachXO3L device family.

Revision 1.4, April 2014

Section	Change Summary
TraceID Access Through the JTAG Port	Updated Table 1, TraceID Register.

Revision 1.3, March 2014

Section	Change Summary
All	Updated Technical Support Assistance information.

Revision 1.2, March 2014

Section	Change Summary
All	• Changed document title to Using TraceID in MachXO2 and ECP5 Devices. • Added support for ECP5 device family.

Revision 1.1, February 2012

Section	Change Summary
All	• Document status changed from advance to final. • Updated document with new corporate logo.
Accessing the TraceID Register	Added the following sections: • TraceID Access Through the JTAG Port • TraceID Access Through the WISHBONE Slave Interface • TraceID Access Through the Slave SPI Port • TraceID Access Through the I^2C Port

Revision 1.0, November 2010

Section	Change Summary
All	Initial release.

www.latticesemi.com