# MachXO3-9400 Development Board I2C Expander for Raspberry Pi Demo

# User Guide

FPGA-UG-02038-A

November 2017

# Contents

# Figures



# Tables

# Acronyms in This Document

A list of acronyms used in this document.

| Acronym | Definition |
|---|---|
| DIP | Dual In-line Package |
| EFB | Embedded Function Blocks. Multiple functions hard blocks in Lattice FPGA |
| GPIO | General Purpose Input and Output |
| GUI | Graphic User Interface |
| $I^2C$ | Inter Integrated Circuit. |
| L-ASC | Lattice Analog Sense and Control. A mixed circuit device of Lattice dedicated for power monitor and management applications. |
| RTL | Register Transfer Level |
| UFM | User Flash Memory |

# 1. Introduction

Many embedded systems are built around microcontrollers or processors with limited I/O and interface. Systems which require more interfaces or additional real time control functions can benefit from the use of a low-density, I/O intensive FPGA such as MachXO3™. This demo shows a basic version of this function using the Raspberry Pi System.

- The demo shows the Lattice MachXO3-9400 Development Board running as an $I^2C$ based GPIO expander to drive the LEDs and a DIP switch on the board and operate the LEDs or read the DIP switch status through a Python based GUI running on the Raspberry Pi 2 or Pi 3. (Section 9)
- The demo shows the Lattice solution to configure the MachXO3 FPGA through an embedded processor. (Section 10)
- The demo shows how to use Raspberry Pi $I^2C$ to operate and to program the Lattice Hardware Management Expander device, L-ASC10. (Section 11)

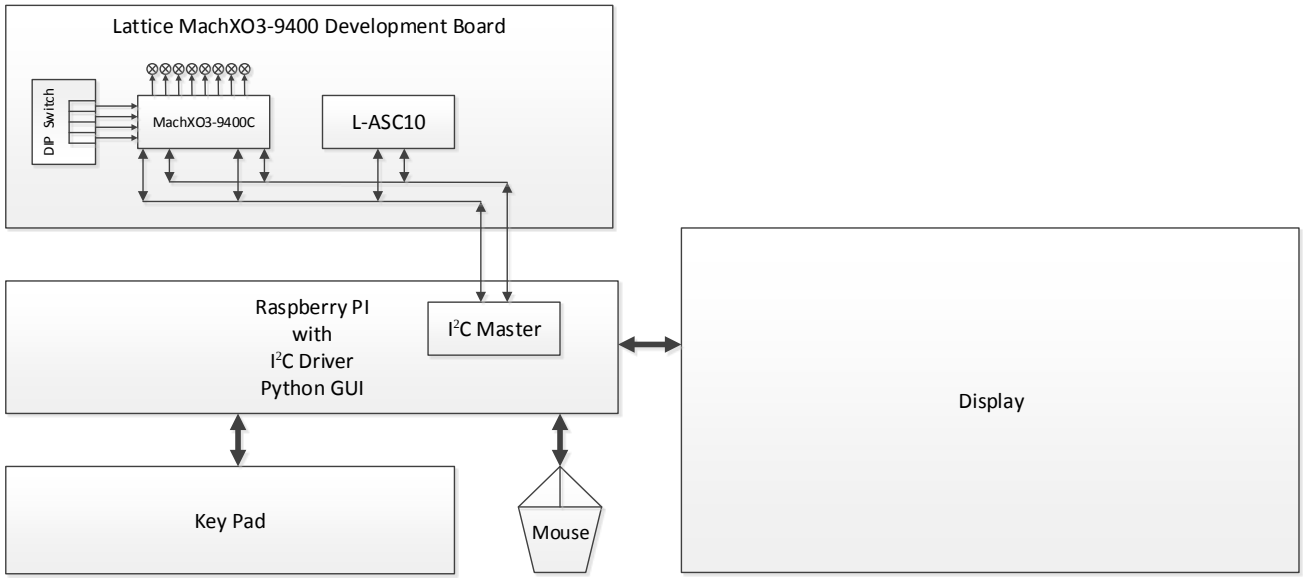


**Figure 1.1. Using Lattice MachXO3-9400 Development Board as Raspberry $I^2C$ Expander**

RTL source code and software source code used with the demo are provided with the demo package.

# 2. Demo Requirements

## 2.1. Hardware Requirements

To demonstrate the simple hardware management design, the following hardware are required:

- MachXO3-9400 Development Board
- Raspberry Pi2/3 running Raspbian
- HDMI display and HDMI display cable
- USB keypad and USB mouse
- Mini USB cable for powering MachXO3-9400 Development Board
- Micro USB cable for powering Raspberry Pi

**Note**: The HDMI display and USB keypad/mouse are not necessary when Windows Remote Desktop tool is used to operate the Raspberry Pi.

## 2.2. Software Requirements

- Lattice Diamond® Design Software, version 3.9 or later
- Lattice Diamond Programmer Tool for bitstream downloading
- Diamond Deployment Tool for $I^2C$ Embedded programming files generation
- Software running on Raspberry Pi:
    - Python based GUI
    - $I^2C$ driver for Python GUI

The Lattice Diamond software tools are available at www.latticesemi.com/en/Products/DesignSoftwareAndIP.
The Programmer tool and Deployment tool are included with Lattice Diamond.

# 3. Setting up the MachXO3-9400 Development Board for the Demo

Figure 3.1 below shows all resources on the MachXO3-9400 Development Board used for the demo.

- JP1 should be set to provide power of L-ASC10 device.
- JP7 should be set to provide +5 V power for Raspberry Pi by the USB download cable. Raspberry Pi can also be powered through a separate Micro USB cable.
- JP3 is a 2.54-cm pitch 2x20 female connector. It is used to connect the MachXO3-9400 Development Board with Raspberry Pi. JP3 is not populated on the board as shipped and required to be mounted for the demo.
- LED D4, D8, D7, D1, D2, D5, D3, D6 are driven by the design software.
- The bits of switch SW1 may be read out by the design software.

**Note**: R84, R96 0 Ω resistors are not populated on the board as shipped. These must be populated to run the demo.

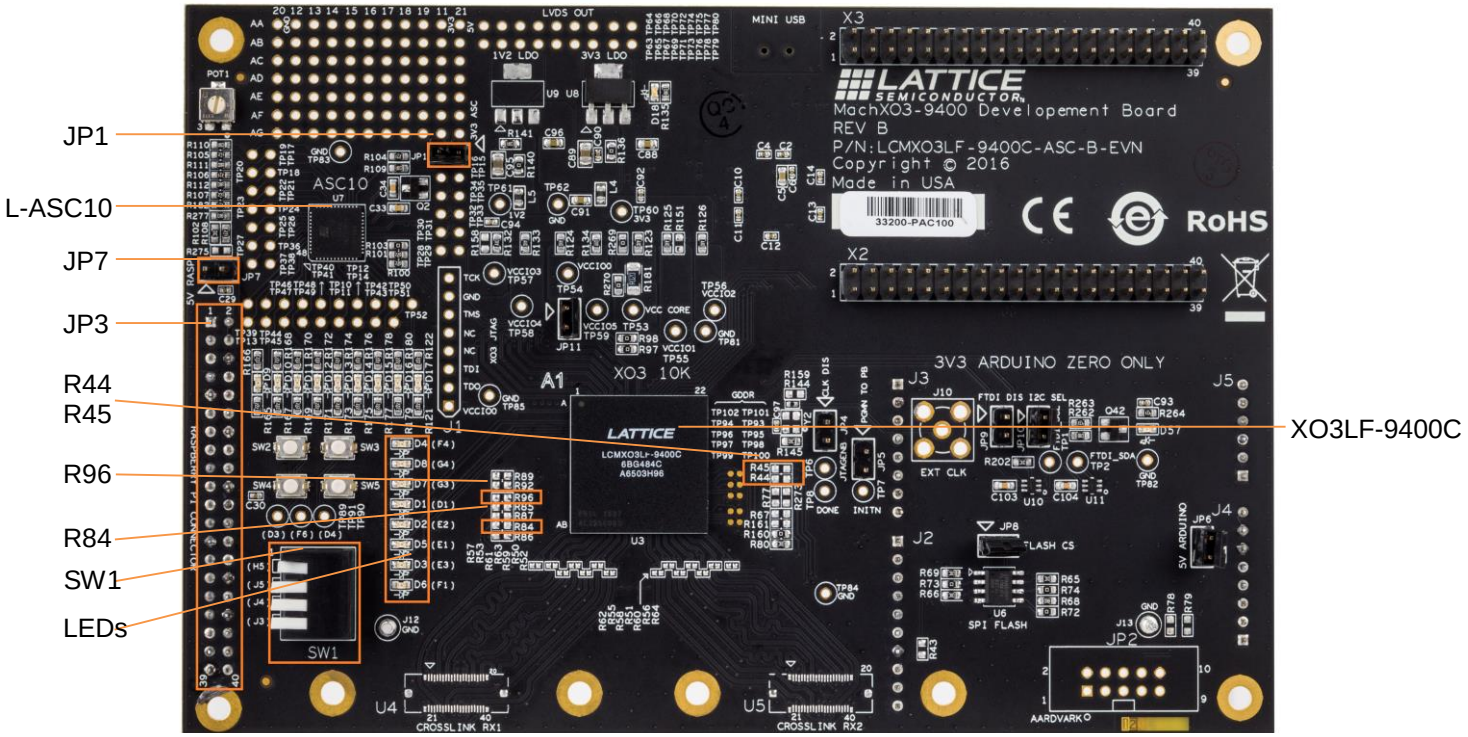


**Figure 3.1. Resources for the Demo**

# 4. Demo Design Overview

A simple design for the MachXO3-9400C device is required for the Demo. The design implementation and software interface for driver are introduced below.

## 4.1. Implementation of I²C Expander using EFB Hard Module in MachXO3 Devices

This section describes how to implement the embedded function block (EFB) in the MachXO3 device. The demo project provides an IPexpress configuration file that includes the settings below. This section is provided for reference only. The EFB is a hardened function block in the MachXO3 device supporting functions such as SPI/I²C/Timer and User Flash Memory (UFM). An 8-bit Wishbone bus interface is used to connect the EFB with user logic. The functions of the block can be configured through Lattice IPexpress software.

To configure the EFB block for the I²C Expander design:

1. Start IPexpress and select the EFB module for configuring.
2. Input the project path, module name, module output format and device family, synthesis tool. Click **Customize** (Figure 4.1).

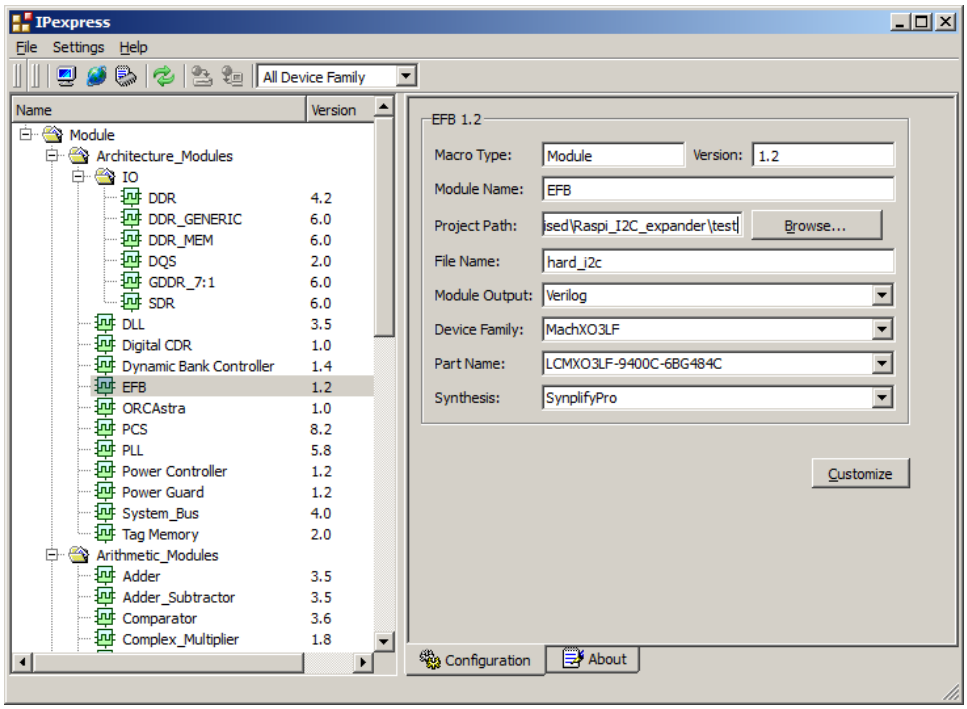


**Figure 4.1. Configuring EFB Module in IPexpress**

3. In the FPGA module dialog box (Figure 4.2), enable the Secondary User I²C and input the Wishbone bus clock frequency leaving other options in defaults. The internal oscillator configured to 53.2 MHz is used as system clock, thus the Wishbone clock is input as 53.2 MHz. Click the **Generate** button. The hard I²C module is generated in the given folder.

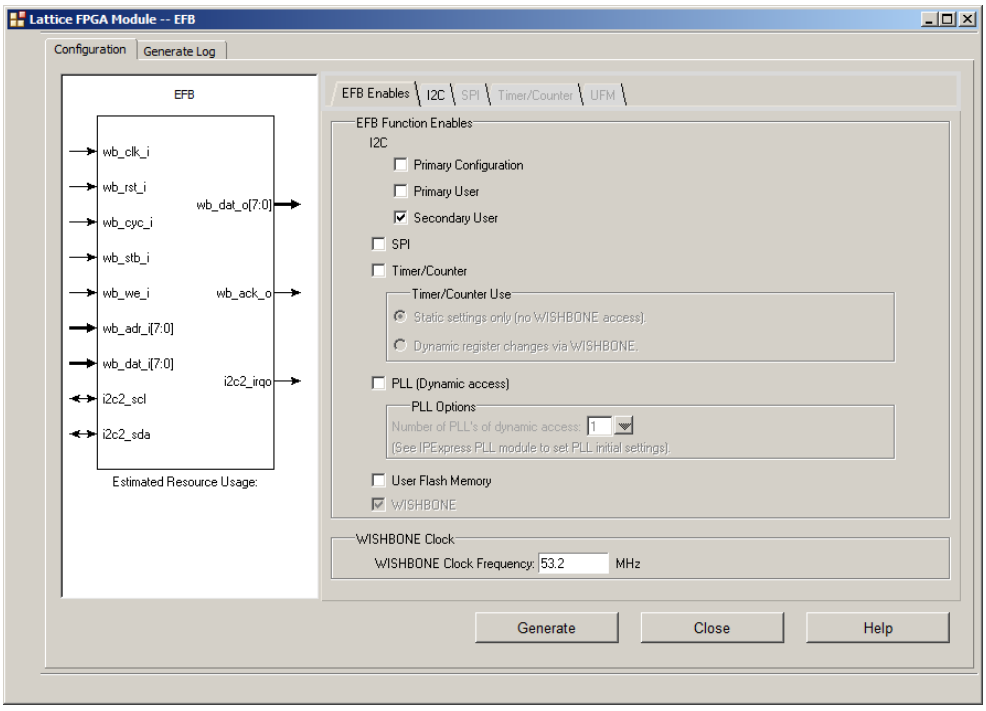


**Figure 4.2. FPGA Module Dialog**

4. Two other modules are provided in the design to implement the I²C expander (Figure 4.3).

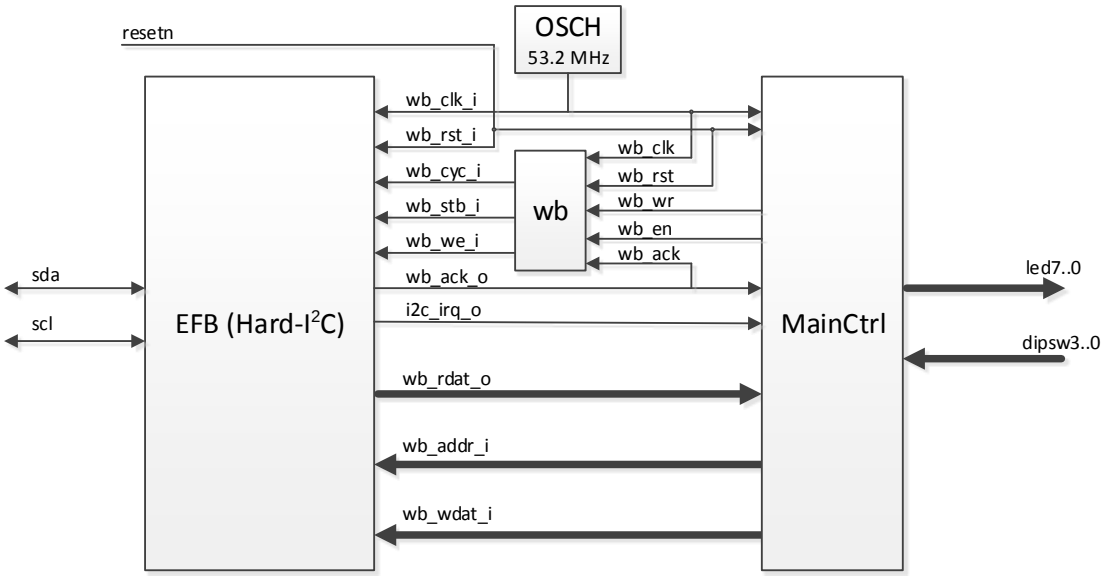


**Figure 4.3. MachXO3-9400C Design Block Diagram**

- The wb module is used to generate Wishbone bus interface signals.
- The MainCtrl module initializes EFB module and updates LEDs based on command and data from the I²C bus. It also provides feedback of the DIP switch status to the EFB module based on the read command.
- All the source files can be found at <*drive:*>/…/source/ directory. The EFB configuration files are at <*drive:*>/…/source/ipexpress/ directory.

## 4.2. Software Interface between Raspberry Pi and MachXO3 I²C Expander Design

The hard I²C of the MachXO3 EFB is configured with the default 7-bit address: 7'b1000010. The 8 LEDs are directly driven by an 8-bit register in the design, with the register address 0x02. The LEDs can be controlled by an I²C write command to register 0x02 at slave address 7'b1000010.

The DIP Switch is connected to an internal 4-bit input register at address 0x01. The switch status can be read out by an I²C read command that reads register 0x01 at slave address 7'b1000010.

The I²C bus of L-ASC10 is directly connected to Raspberry Pi I²C master port. The I2C_ADDR pin of the device is connected to GND through a 0 Ω resistor, R103, so the 7-bit I²C slave address of the device is 7'b1100000. Registers in the devices can directly be accessed by the Raspberry Pi through the I²C Bus.

# 5. MachXO3 Design – Port Assignments and Descriptions

Table 5.1 shows all ports definitions in the design.

**Table 5.1. Design Port Definitions**

| Port Name | Direction | Description |
|---|---|---|
| resetn | Input | External Reset input from Raspberry Pi, active low. In applications related to the L-ASC10 device, Raspberry Pi must hold the resetn active until the end of L-ASC10 programming. |
| scl | Inout | $I^2C$ bus clock |
| sda | Inout | $I^2C$ bus data |
| dipsw[3:0] | Input | 4-bit DIP Switch SW1 for the $I^2C$ expander to read |
| led[7:0] | Output | LEDs (D4, D8, D7, D1, D2, D5, D3, D6) for the $I^2C$ expander to drive |

# 6. Demo Package Components and Directory Structure

This demo package includes the following components:

- Verilog source code for the demo logic design
- Lattice Diamond® Project file and preference file for the demo project
- $I^2C$ based GPIO Expander design JEDEC file for programming XO3 internal configuration flash
- Python based GUI for operating the LEDs, reading DIP switch, programming XO3-9400C and ASC-10 device
- $I^2C$ driver C code for Python GUI to operate the $I^2C$ bus
- C code for Lattice $I^2C$ embedded programming solution based on the Raspberry Pi board
  - The general ispVM Embedded C code can be found in the Diamond installation directory <*drive:*> /lscc/diamond/3.9(_64)/embedded_source/i2cembedded/src/i2cem
  - The updated C code for Raspberry Pi 2 can be found in the demo installation directory /Raspi_I2C_expander/raspi/ispvm_embedded/c

Figure 6.1 shows the demo design package directory structure. The design package is available as a zip file on the website on the same page as for this document. You can download it and extract to your local disk.

After the ZIP is extracted, you can see the folder structure as shown in Figure 6.1. Major components in the folder structure are described as follows:

- **Raspi_I2C_expander** is the root directory.
- **bitstream** contains the JEDEC file (XO3 FPGA configuration) and HEX file (L-ASC10 configuration) for the demonstration
- **docs** (contains this document)
- **hardware**
  - **implementation** contains Diamond project file
    - **impl1**
  - **source** contains HDL files for the demonstration
- **raspi** contains tools running on the Raspberry Pi for the demo

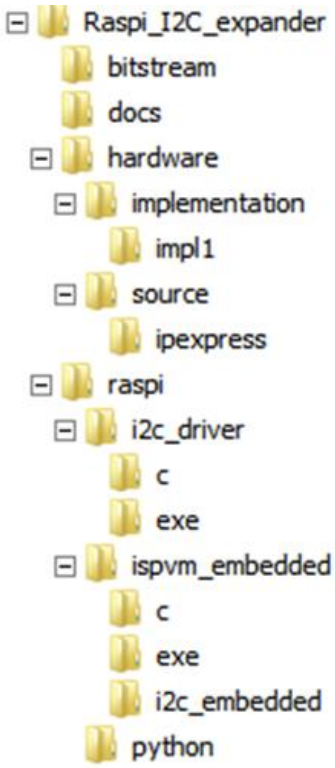


**Figure 6.1. Demo Design Package Directory Structure**

# 7. Programming the MachXO3-9400C Device

Before starting the Demo, you need to use Diamond Programmer to program the MachXO3-9400C device on your PC.

You can program the MachXO3-9400C device on your PC following the steps below:

1. Make sure Raspberry Pi is unconnected or powered off.
2. Launch the Diamond Programmer Software, version 3.9 or above. In the Getting Started dialog box (Figure 7.1), select **Create a new project from a JTAG scan** and click **OK**.

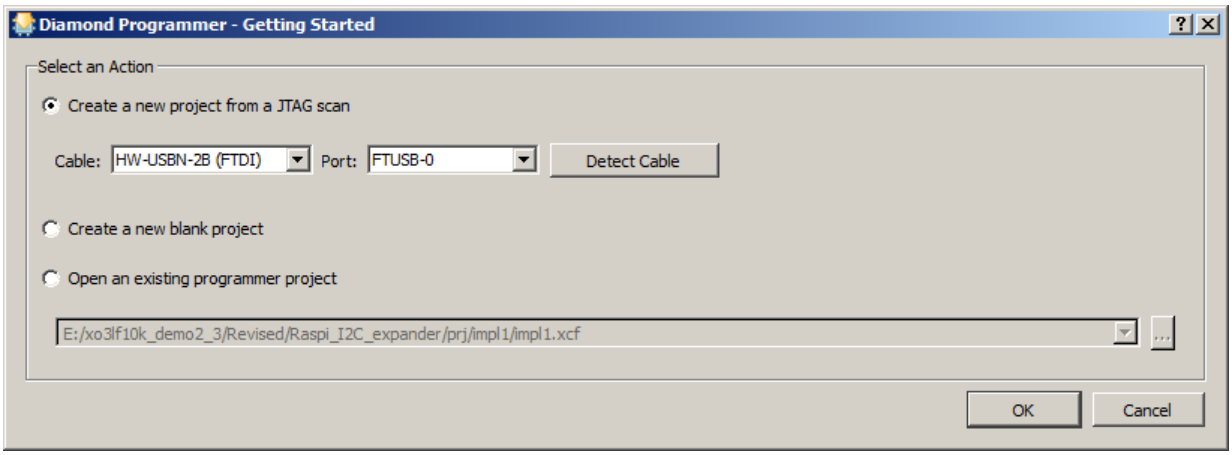


**Figure 7.1. Diamond Programmer Getting Started Dialog**

3. The Diamond Programmer starts scanning the board attached to the USB cable (Figure 7.2).

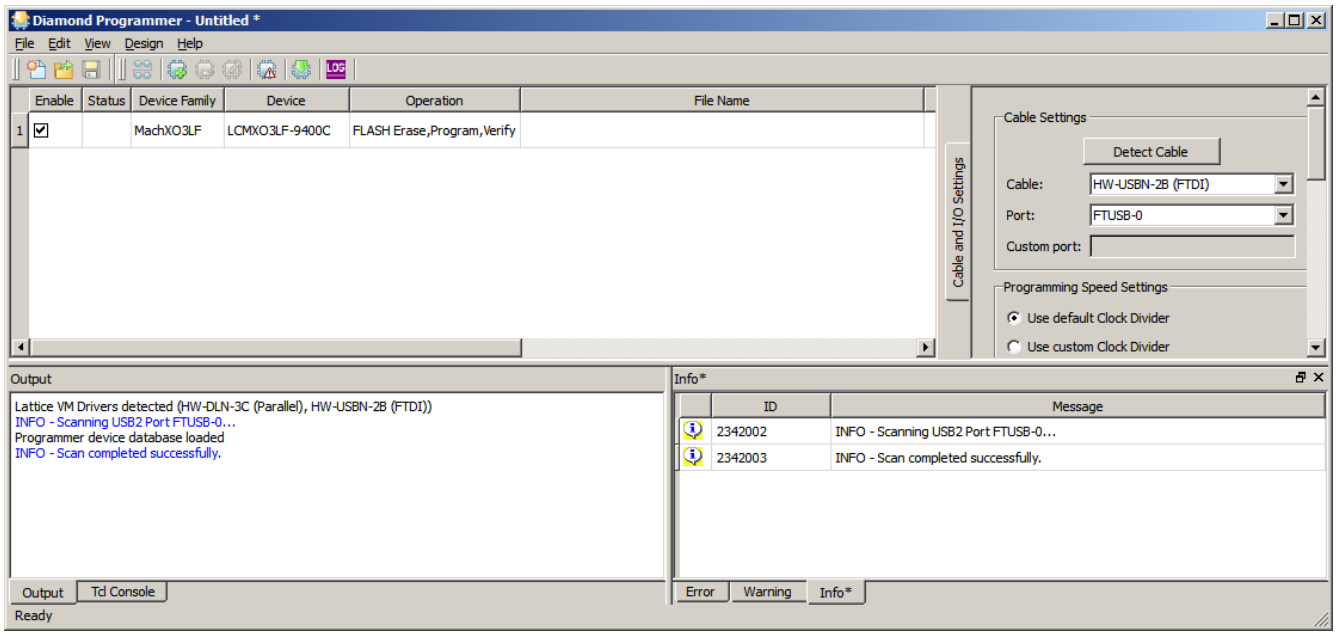


**Figure 7.2. Diamond Programmer Main Interface**

4. Click and highlight the device row. Select **Edit > Device Properties** from the menu bar (Figure 7.3).

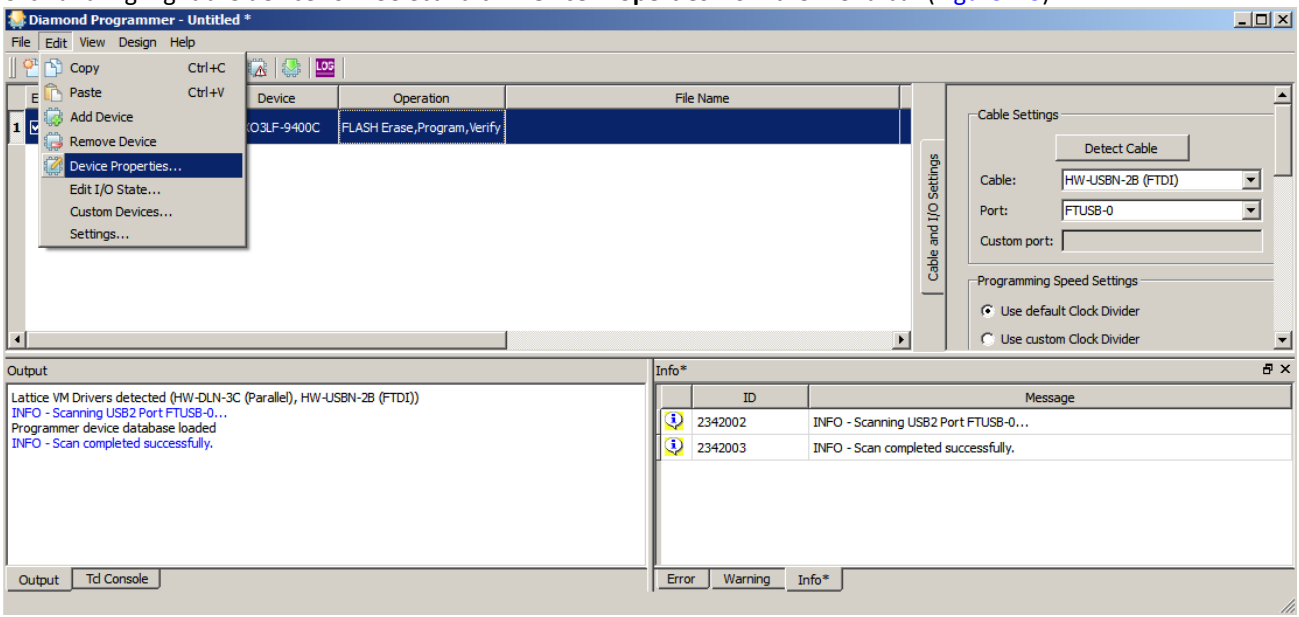


**Figure 7.3. Editing Device Properties**

5. The Device Properties dialog pops up (Figure 7.4). You can edit the Access mode, Operation mode and select the programming file.

   For this Demo, in the Device Properties dialog (Figure 7.4), select **Flash Programming Mode** in the Access mode field, and **Flash Erase, Program, Verify** in the Operation field. Select **raspi_i2c_expander_impl1.jed** as the Programming file. Click **OK**.

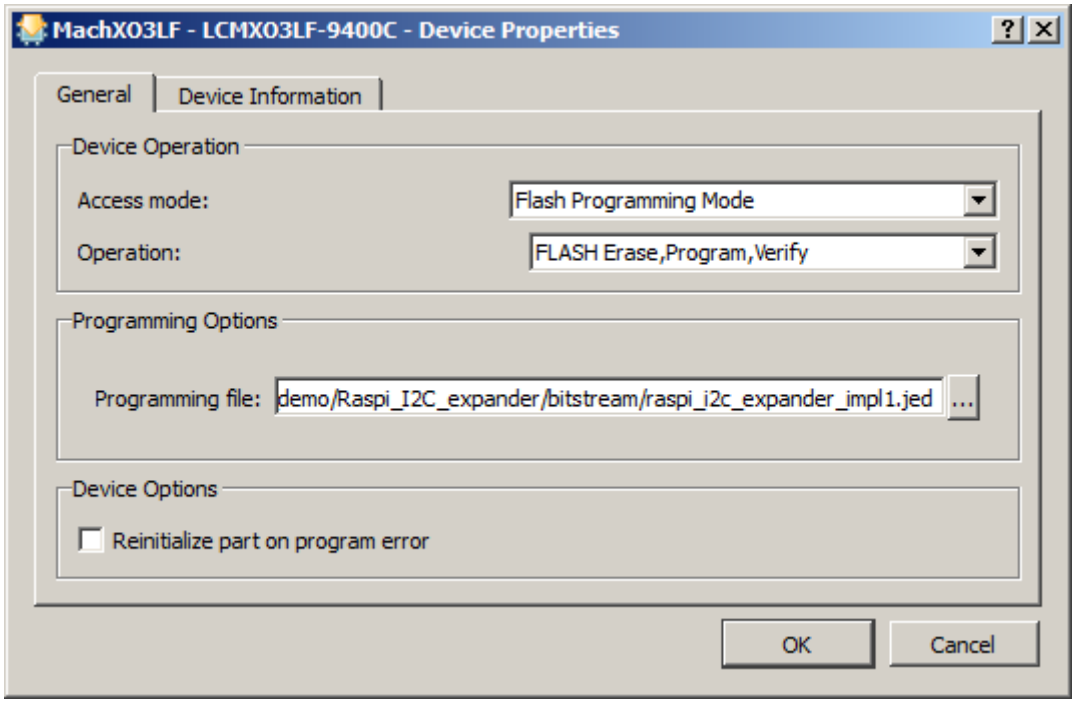


**Figure 7.4. Selecting Device Operation and Programing Options**

6. From the menu bar of the Diamond Programmer, select **Design > Program** (Figure 7.5). The XO3LF-9400C is programmed.

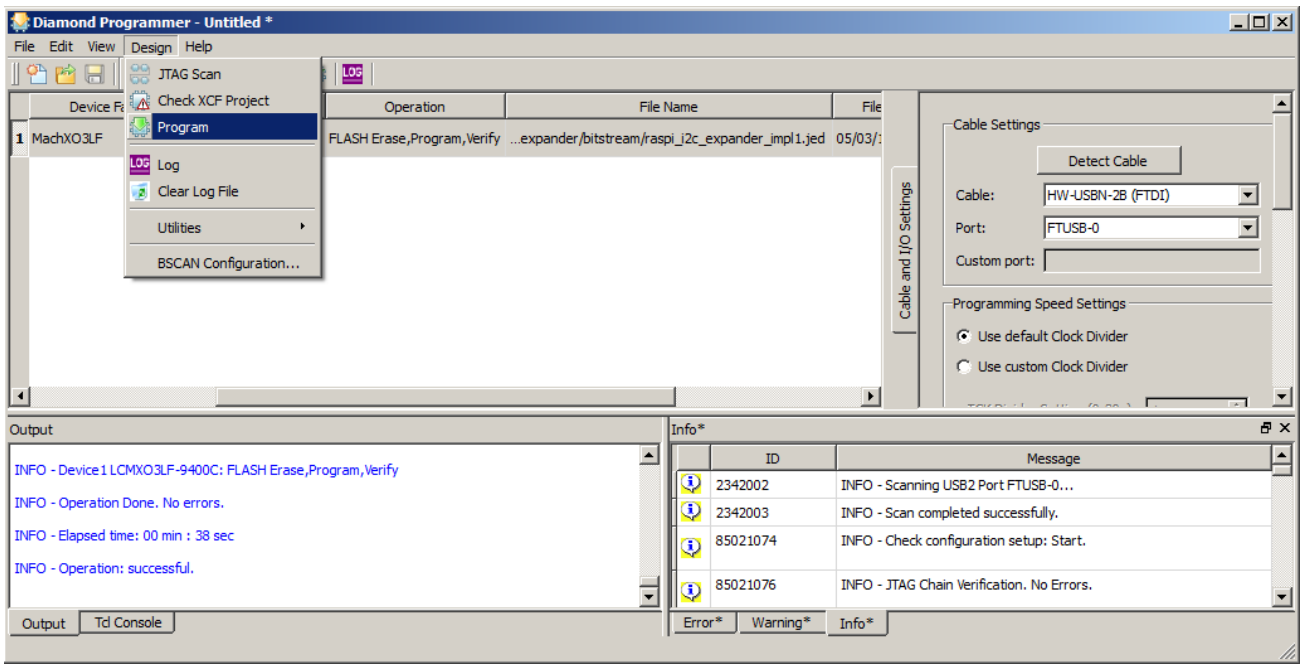


**Figure 7.5. Programming the Device**

If no error is reported, the MachXO3-9400 device is programmed successfully with the design.

# 8. Prepare the Demo Environment on the Raspberry Pi

Before starting your Demo, follow the steps below to prepare the demo environment on the Raspberry Pi:

1. Connect the Raspberry Pi with the MachXO3-9400 Development Board through J8 on the Raspberry Pi and JP3 on the MachXO3-9400 Development Board. Make sure the pin numbers on both connectors are consistent before connecting them.
2. Connect USB keypad, USB mouse, and HDMI display to Raspberry Pi. Power up the display.
3. Power up Raspberry Pi.

   **Note**: This demo can be run on either Raspberry Pi 2 or Raspberry Pi 3.
4. Install software for FTP support and BCM2835 SPI/I2C/GPIO support library onto Raspberry Pi.
5. Copy all folders, files under /Raspi_I2C_expander/raspi in the design package from your PC to the Raspberry Pi /home/pi directory using the FTP Client tool.
6. From the Raspberry Pi console, add executable attribute to all executable files at /home/pi/raspi/i2c_driver/exe/ using this command:

```
chmod +x /home/pi/raspi/i2c_driver/exe/*
```

7. Files under /home/pi/raspi/ are used to complete the three Demos in Section 9, Section 10, Section 11 through the I2C Expander and Programmer GUI (Figure 8.1).

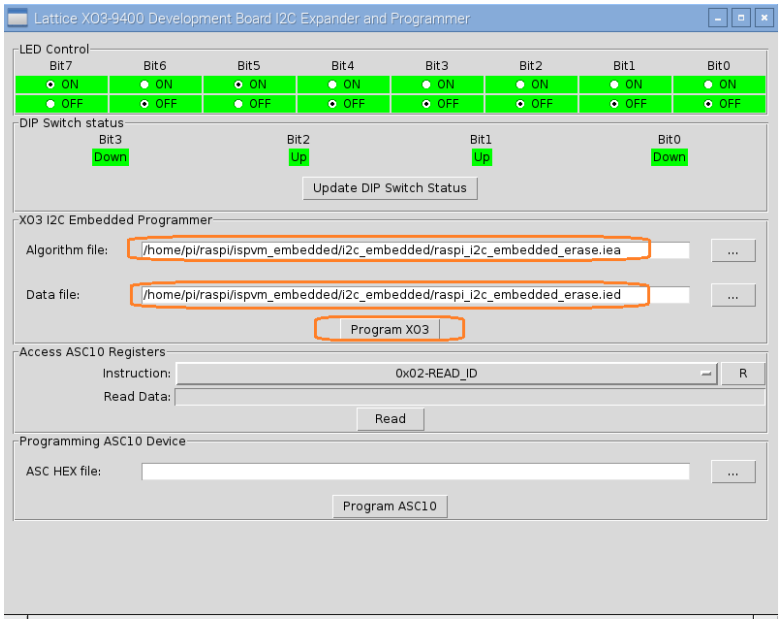


**Figure 8.1. I2C Expander and Programmer GUI**

# 9. Running the I²C Expander Demo on Raspberry Pi

Now you can run the I2C Expander demo on Raspberry Pi.

1. Start Python Shell from Raspberry Pi by choosing **Menu > Programming > Python 2** (Figure 9.1).

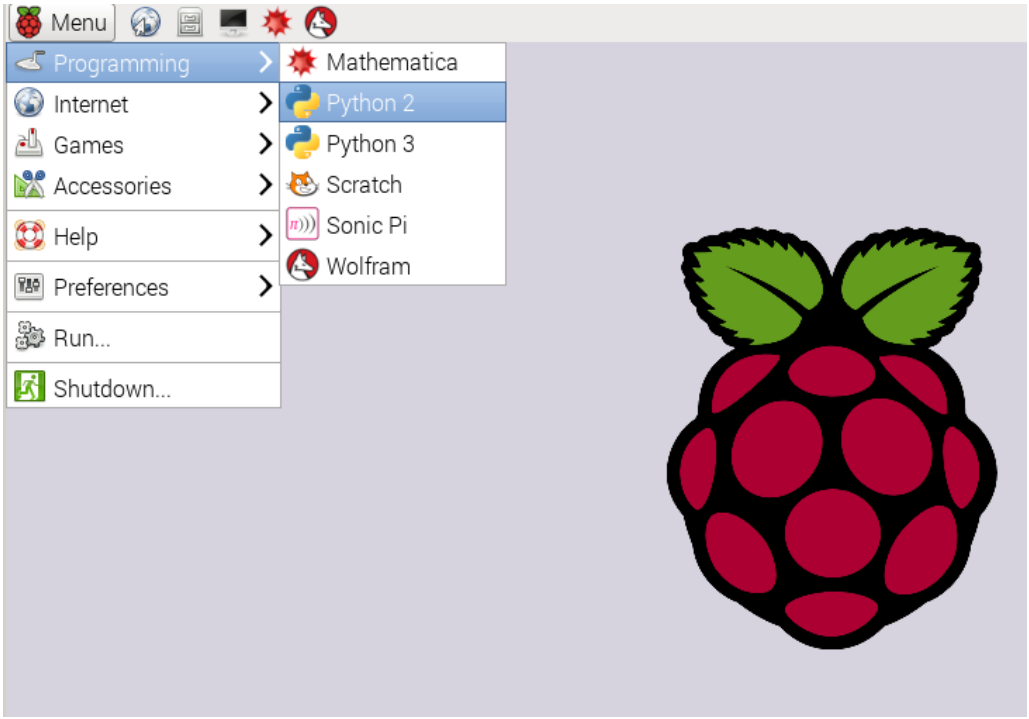


**Figure 9.1. Starting Python2 IDE**

2. Load the I²C Expander GUI by opening /home/pi/raspi/python/raspi_i2c.py from Python Shell (Figure 9.2).

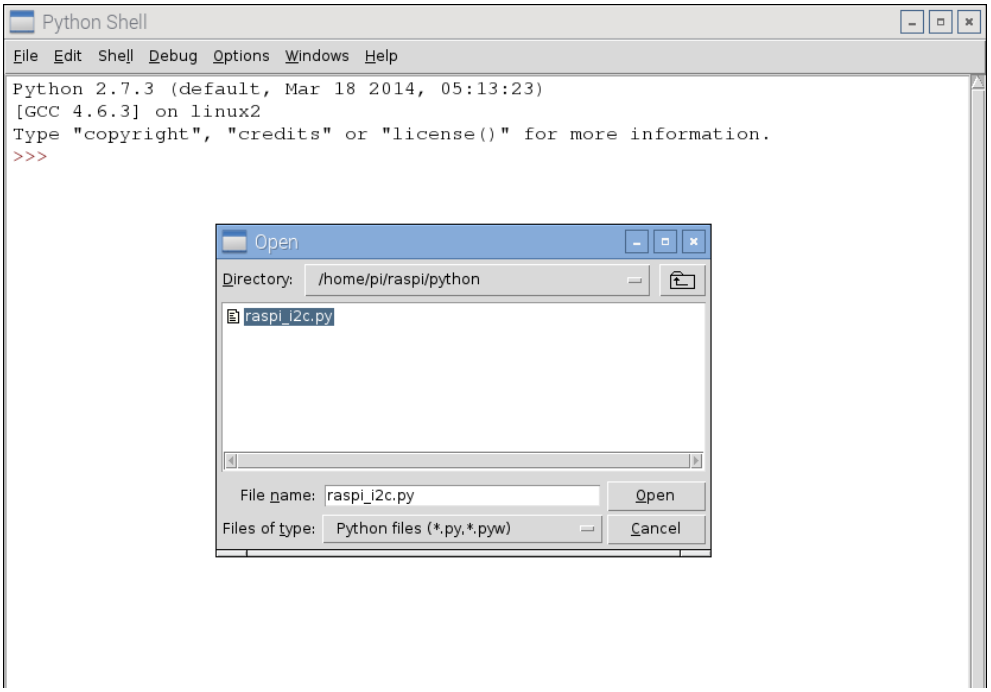


**Figure 9.2. Load the I²C Expander GUI**

3. Start I²C Expander GUI by clicking **Run > Run Module** (Figure 9.3).

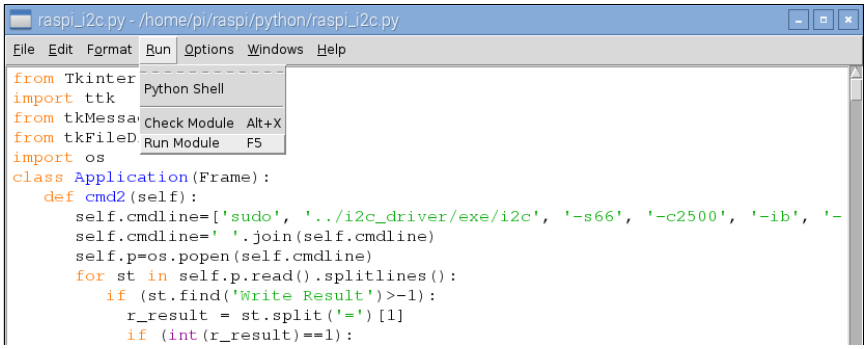


**Figure 9.3. Starting the I²C Expander GUI**

4. Click the green **ON** buttons of Bit7 and Bit4 in GUI (Figure 9.4). The corresponding LEDs on the board (Figure 9.5) turn on.
5. Push down 2 bits of the DIP Switch SW1 on the board (Figure 9.5). Press the **Update DIP Switch Status** button in the GUI (Figure 9.4). You can see the corresponding bits, Bit 3 and Bit 0, (as shown in Figure 9.4) change to show **Down**.

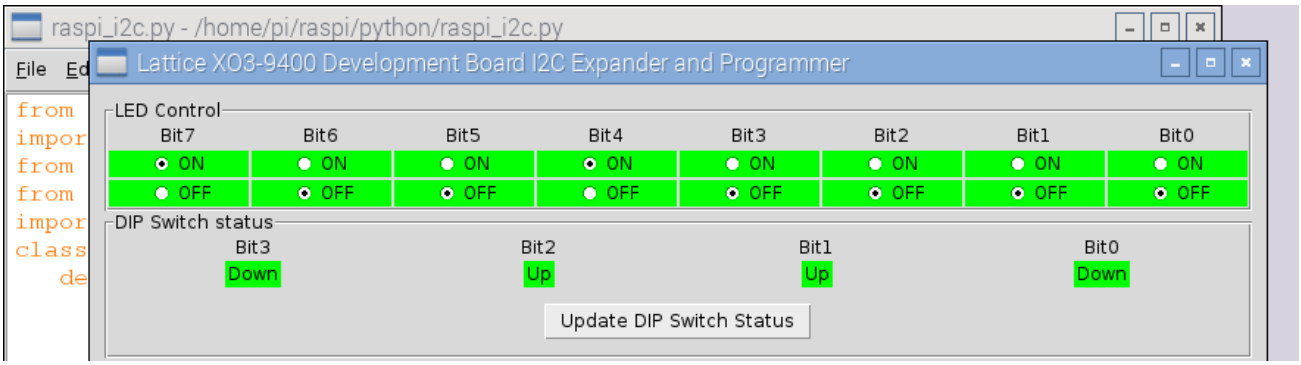


**Figure 9.4. Operate the LEDs and Read DIP Switch Status through the I²C Expander GUI**

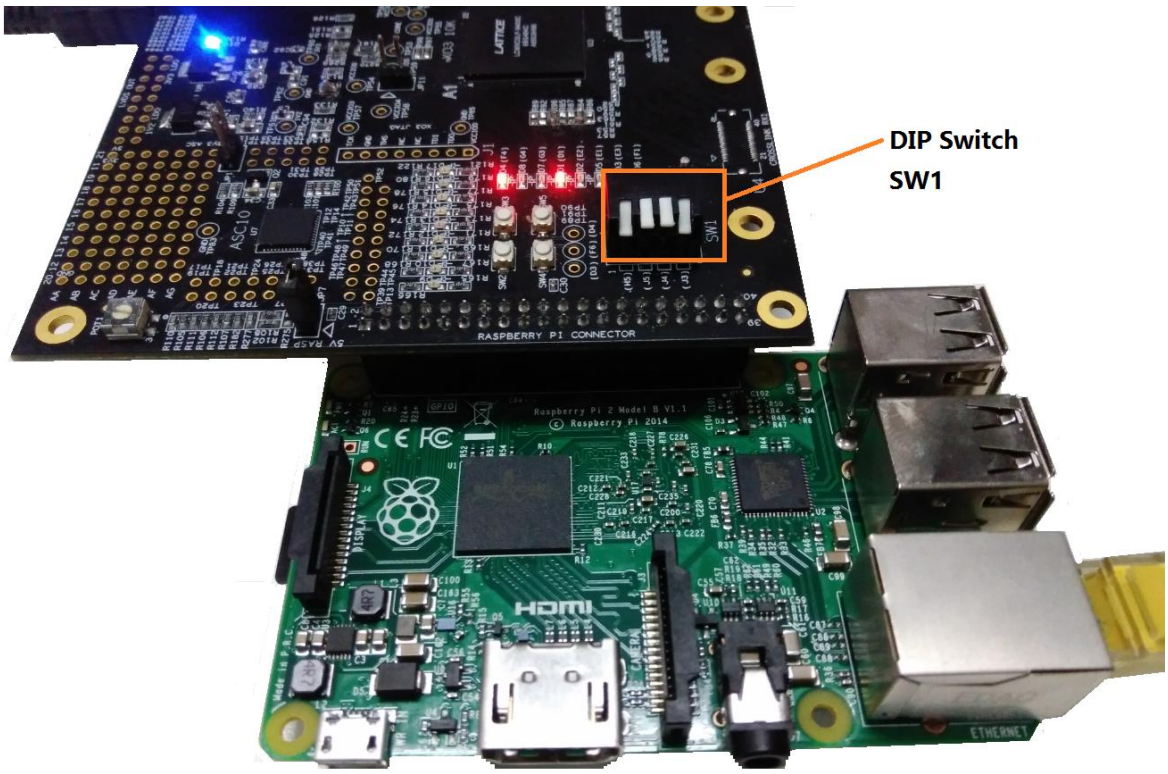


**Figure 9.5. LEDs and DIP Switch Status on the MachXO3-9400 Development Board**

SW1 status is read back through the $I^2C$ Expander.

# 10. Running the I²C Embedded Programming Demo on Raspberry Pi

Here is the Demo showing you how to use Raspberry Pi to program MachXO3-9400C device with Lattice ispVM Embedded programming technology. ispVM Embedded requires special programming files to be generated from the JEDEC file. These special programing files for the demo are pre-generated and enclosed under /Raspi_I2C_expander/raspi/ispvm_embedded /i2c_embedded directory.

If you want to use a new design to test the programming procedure, the special programming files are required to be re-generated following the steps below.

To generate the programing files from the JEDEC file:

1. From your PC, start Diamond Deployment Tool. Select **Create New Deployment** from the Getting Started dialog (Figure 10.1). Select **Embedded System** as the Function Type, and **I2C Embedded** as the Output File Type. Click **OK**.

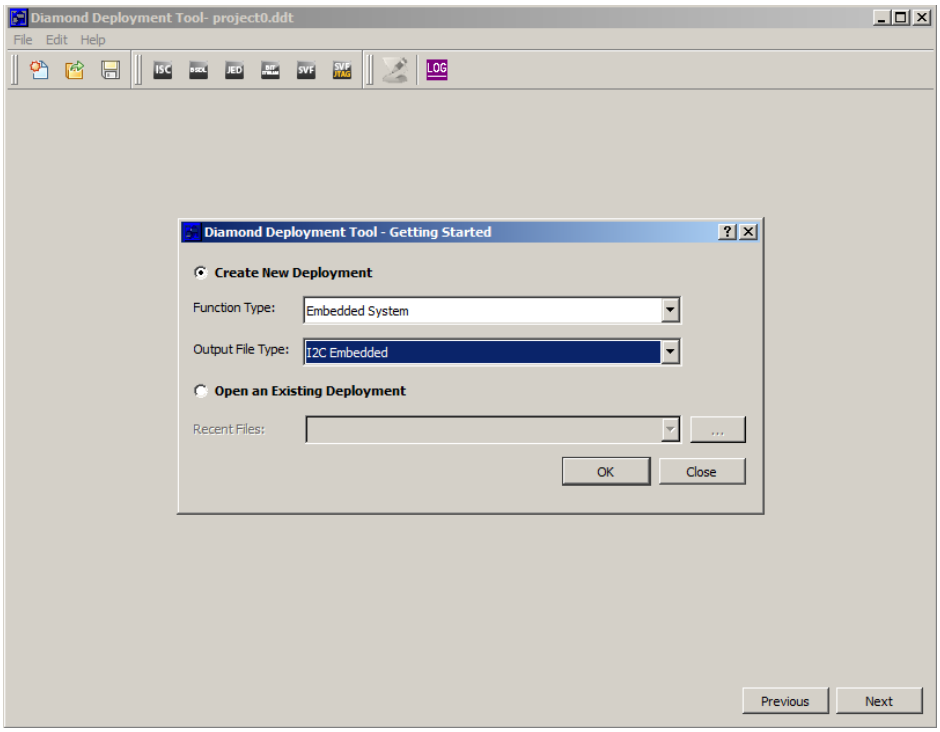


**Figure 10.1. Starting Diamond Deployment Tool for I²C Embedded File Generation**

2. Select a JEDEC file under …/bitstream folder (Figure 10.2). Click **Next** to continue.

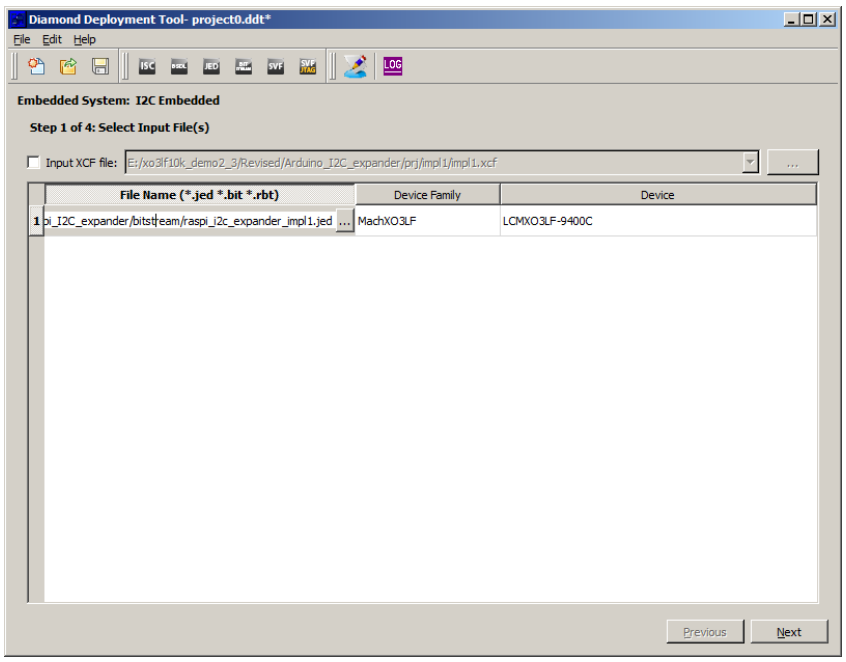


**Figure 10.2. Selecting JEDEC File**

3. Select **I2C Erase Only** as the operation (Figure 10.3). Click **Next**.

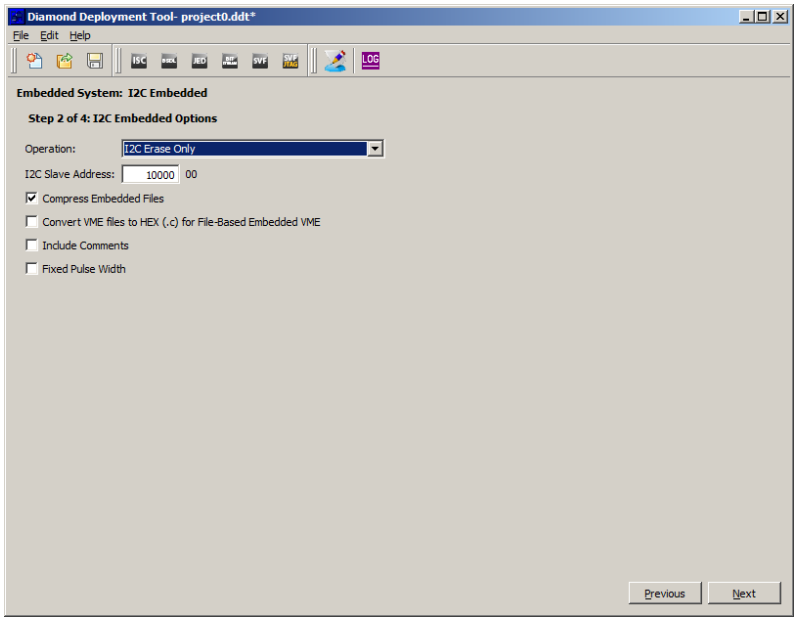


**Figure 10.3. Selecting I2C Erase Only Option**

4. Give a name respectively to the Algorithm File and Data File (Figure 10.4), for example, raspi_I2C_embedded_erase.iea and raspi_I2C_embedded_erase.ied. Click **Next** to continue.

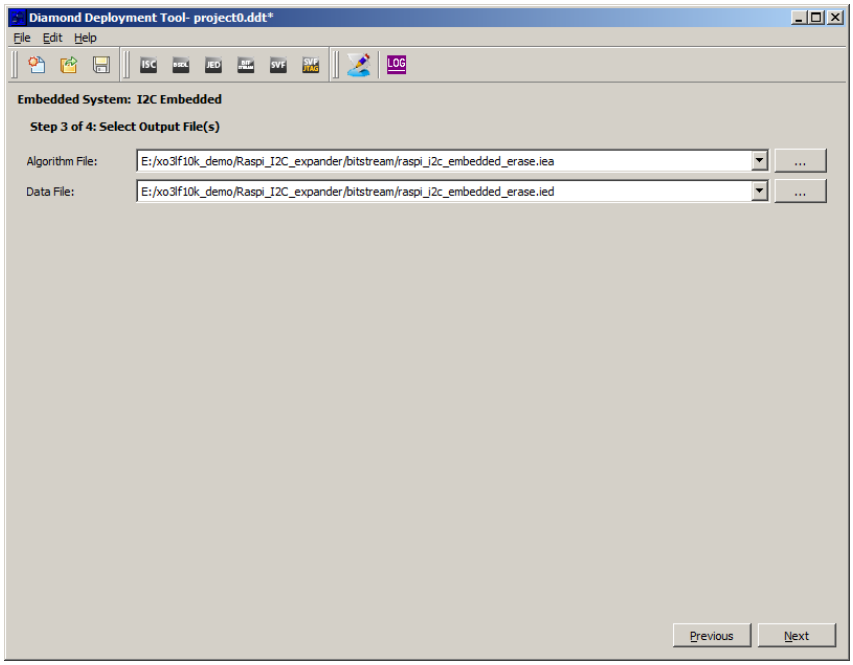


**Figure 10.4. Specifying Names for the Algorithm File and Data File**

5. Click **Generate** to generate the Algorithm file and the data file for the Erase Only operation of the I²C Embedded programming (Figure 10.5).

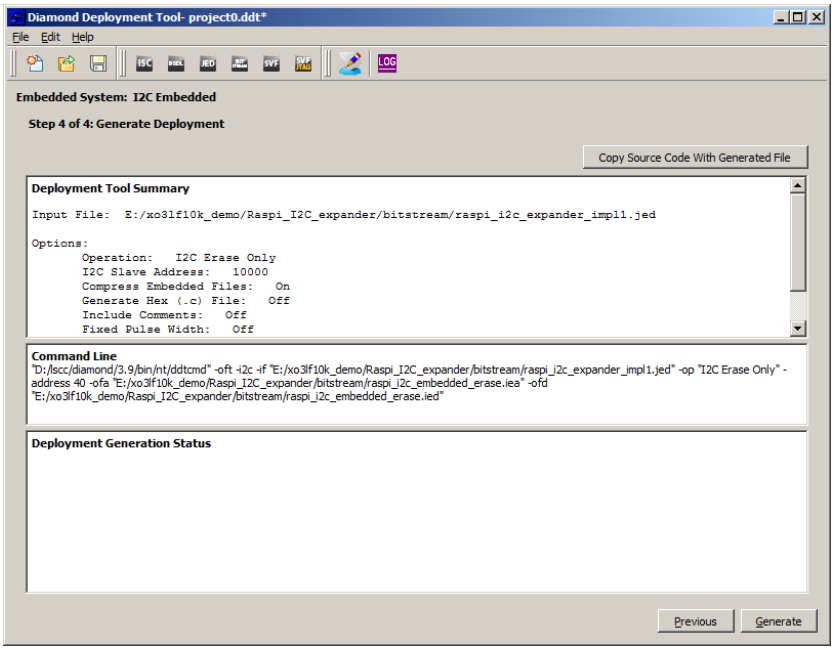


**Figure 10.5. Generating the Algorithm File and Data File**

6. After the file generation completed successfully, you need to regenerate the Algorithm file and the data file for programming. You need to select I2C Program option (as shown in Figure 10.6).

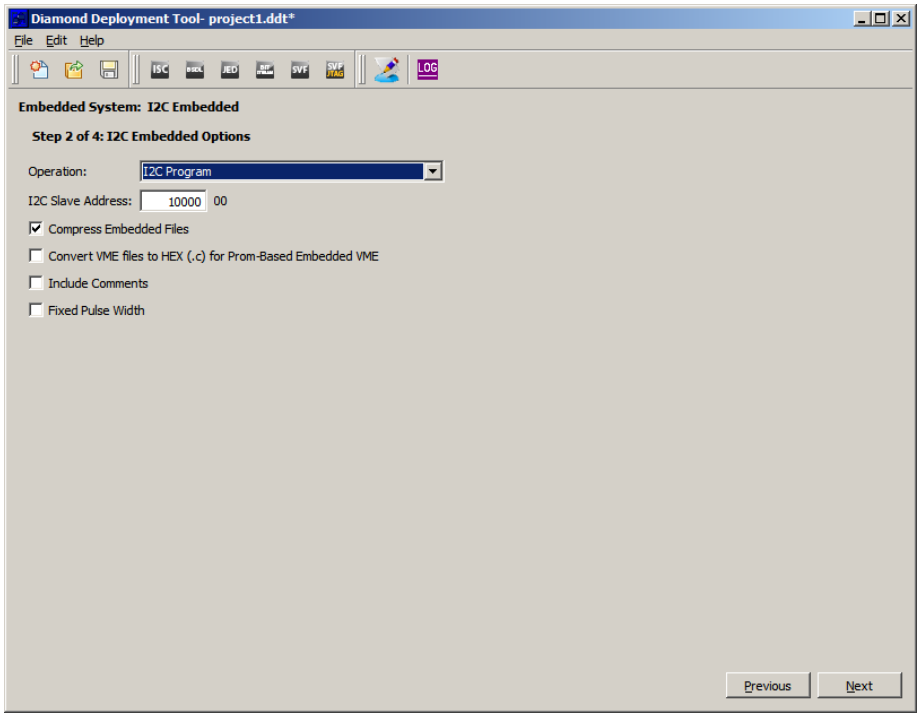


**Figure 10.6. Selecting I2C Program Option**

7. Give a new name respectively to the Algorithm File and Data File (Figure 10.7), for example, raspi_I2C_embedded_program.iea and raspi_I2C_embedded_program.ied. Click **Next** to continue.

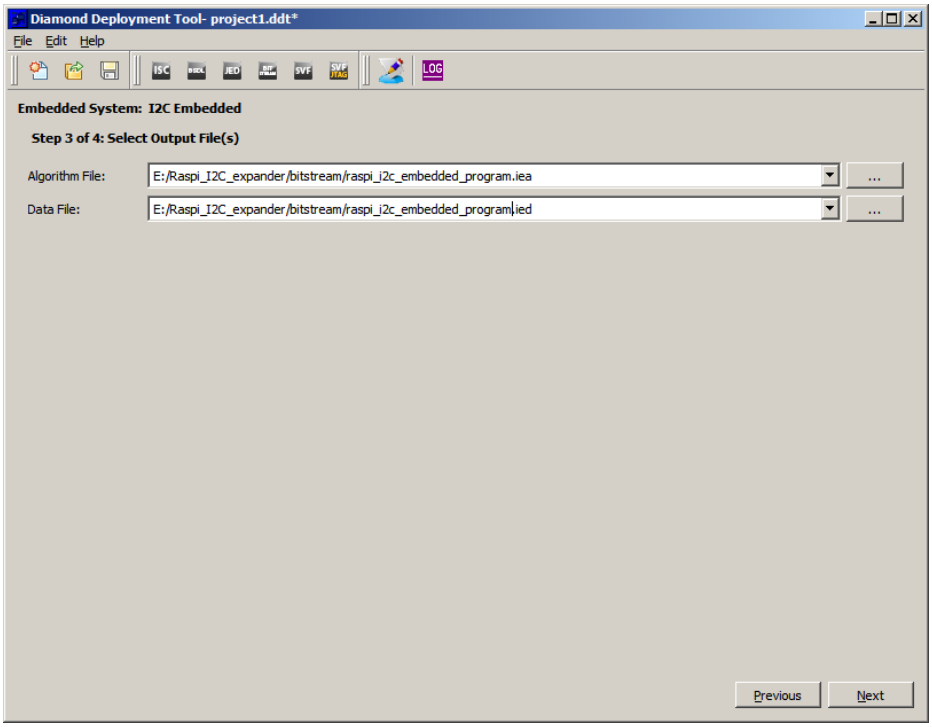


**Figure 10.7. Specifying New Names for the Algorithm File and Data File**

8. Click **Generate** to re-generate the Algorithm file and the data file for the Program Only operation of the $I^2C$ Embedded programming (Figure 10.8).

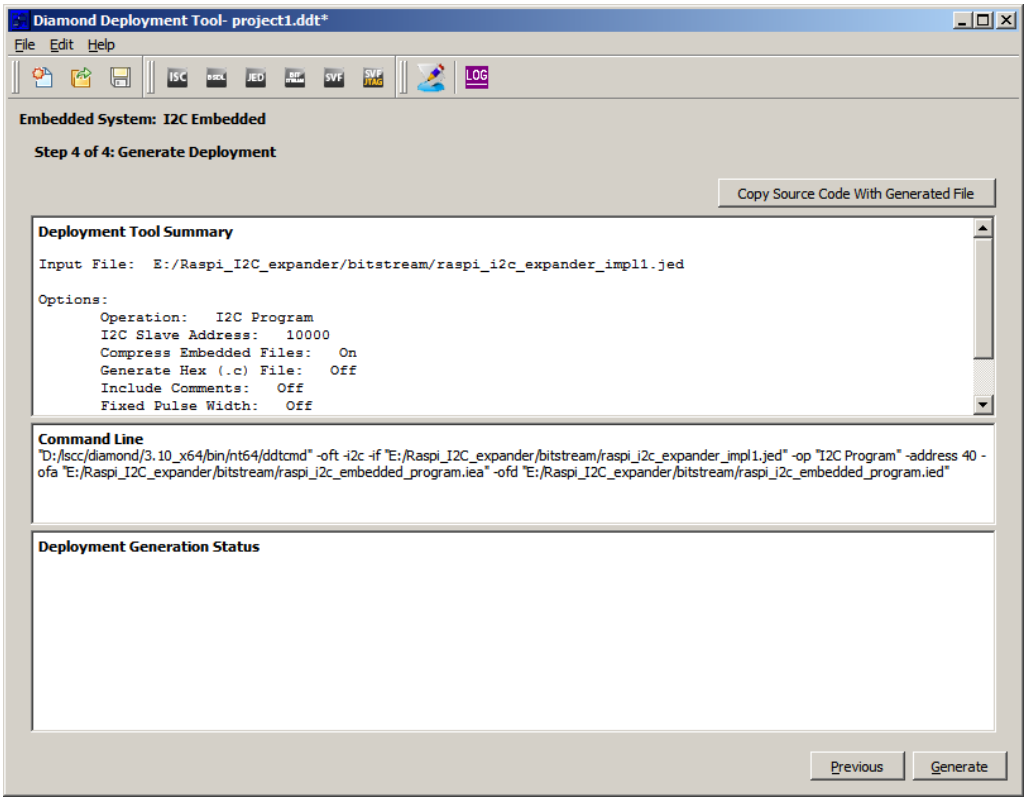


**Figure 10.8. Re-generating the Algorithm File and Data File**

The Diamond Deployment Tool generates the Algorithm file and the data file successfully (Figure 10.9).

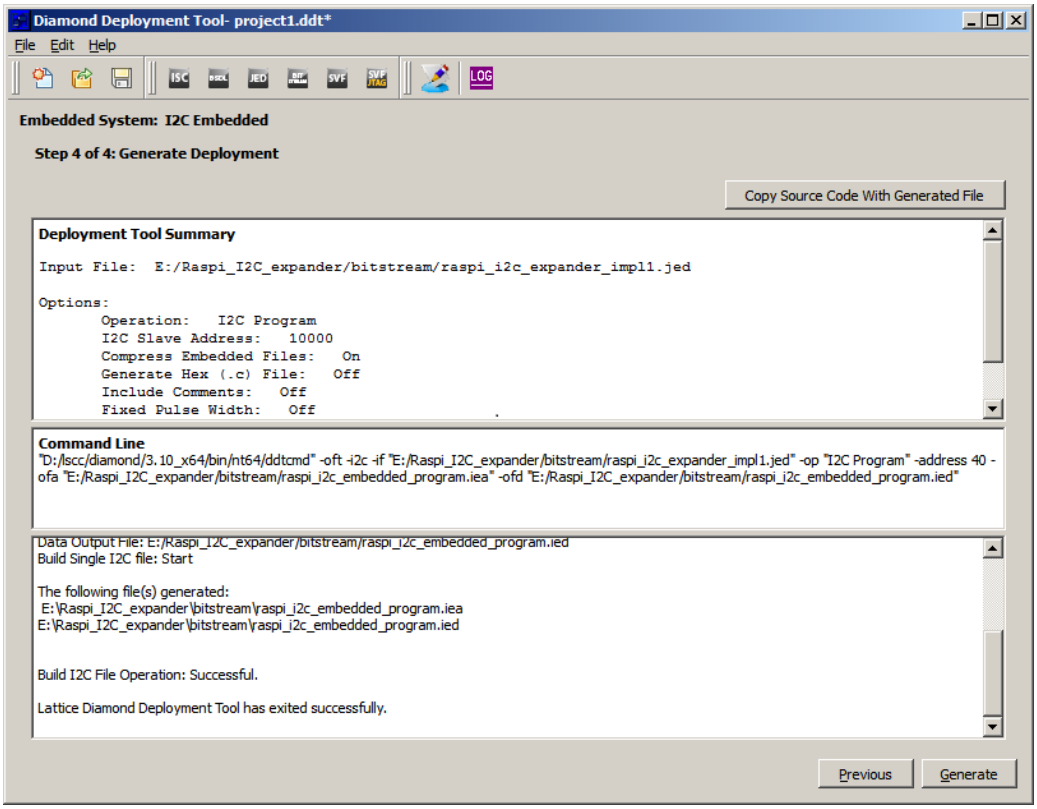


**Figure 10.9. Generating Algorithm File and Data File Successfully**

9. Use the FTP client tool to copy the four generated files to the Raspberry Pi /home/pi /raspi/ispvm_embedded/i2c_embedded directory.

10. On the Raspberry Pi, in the I²C Expander and Programmer GUI, select raspi_I2C_embedded_erase.iea from the Algorithm File field and raspi_I2C_embedded_erase.ied from the Data file field to erase the device. Click the **Program XO3** button (Figure 10.10).

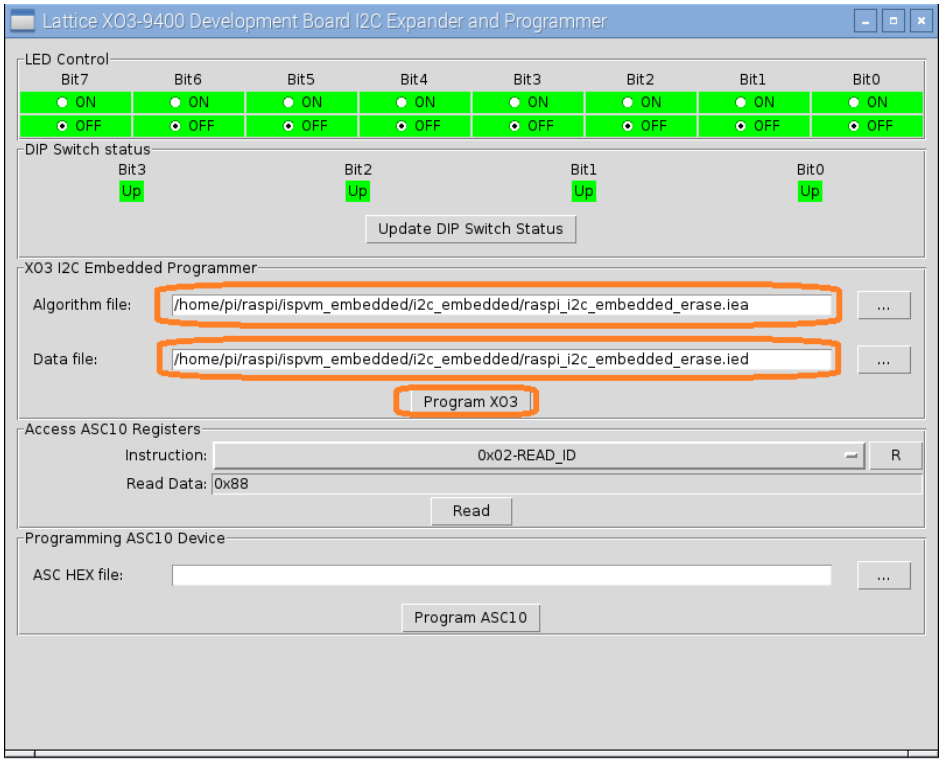


**Figure 10.10. Using Raspberry to Erase XO3**

11. The MachXO3 device is erased. If you click the **Update DIP Switch Status** button again, an error is reported indicating that the device is erased (Figure 10.11).

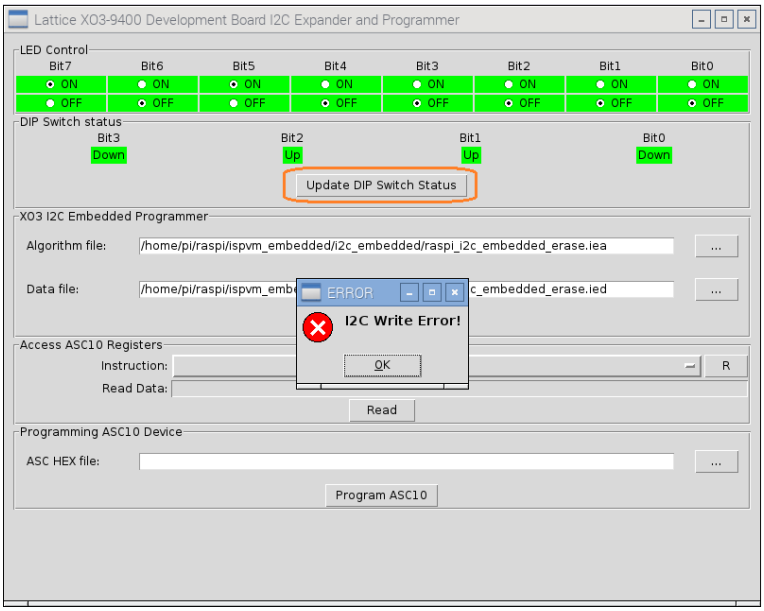


**Figure 10.11. I²C Expander and Programmer Operation Error after the Device is Erased**

12. In the I²C Expander and Programmer GUI, select raspi_I2C_embedded_program.iea from the Algorithm file Field and raspi_I2C_embedded_program.ied from the Data file field to reprogram the device. Click **Program XO3** button (Figure 10.12). The programming process takes about 90 seconds.

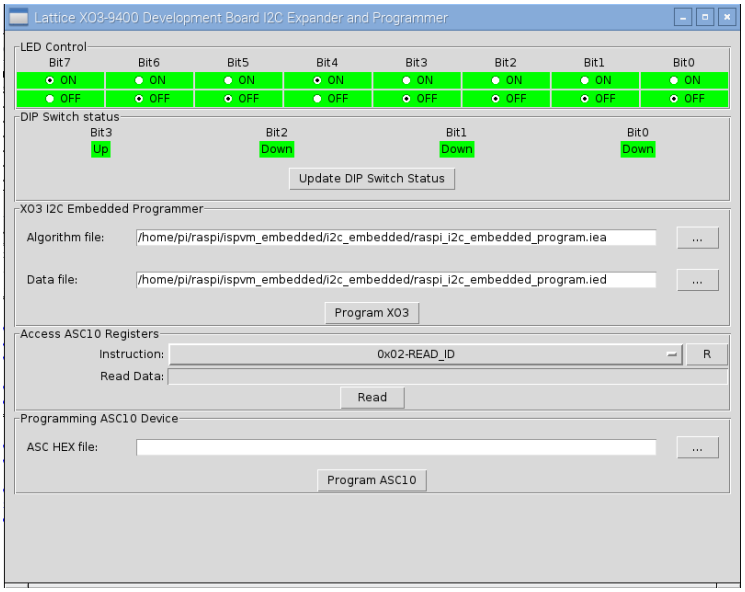


**Figure 10.12. I²C Expander Functioning after XO3 is Reprogrammed by Raspberry Pi**

13. After programming is completed, the LEDs can be set and the DIP Switch status can be read again using the GUI (Figure 10.12).

# 11. Running the L-ASC10 Device Programming over I²C Demo on Raspberry Pi

The Python based I²C Expander and Programmer GUI supports all L-ASC10 I²C commands to access the device internal resources. Supported commands can be selected in a pull down list box as shown below (Figure 11.1) for accessing the L-ASC10 device.

See L-ASC10 In-System Programmable Hardware Management Expander Data Sheet (DS1042) for details on the command set.

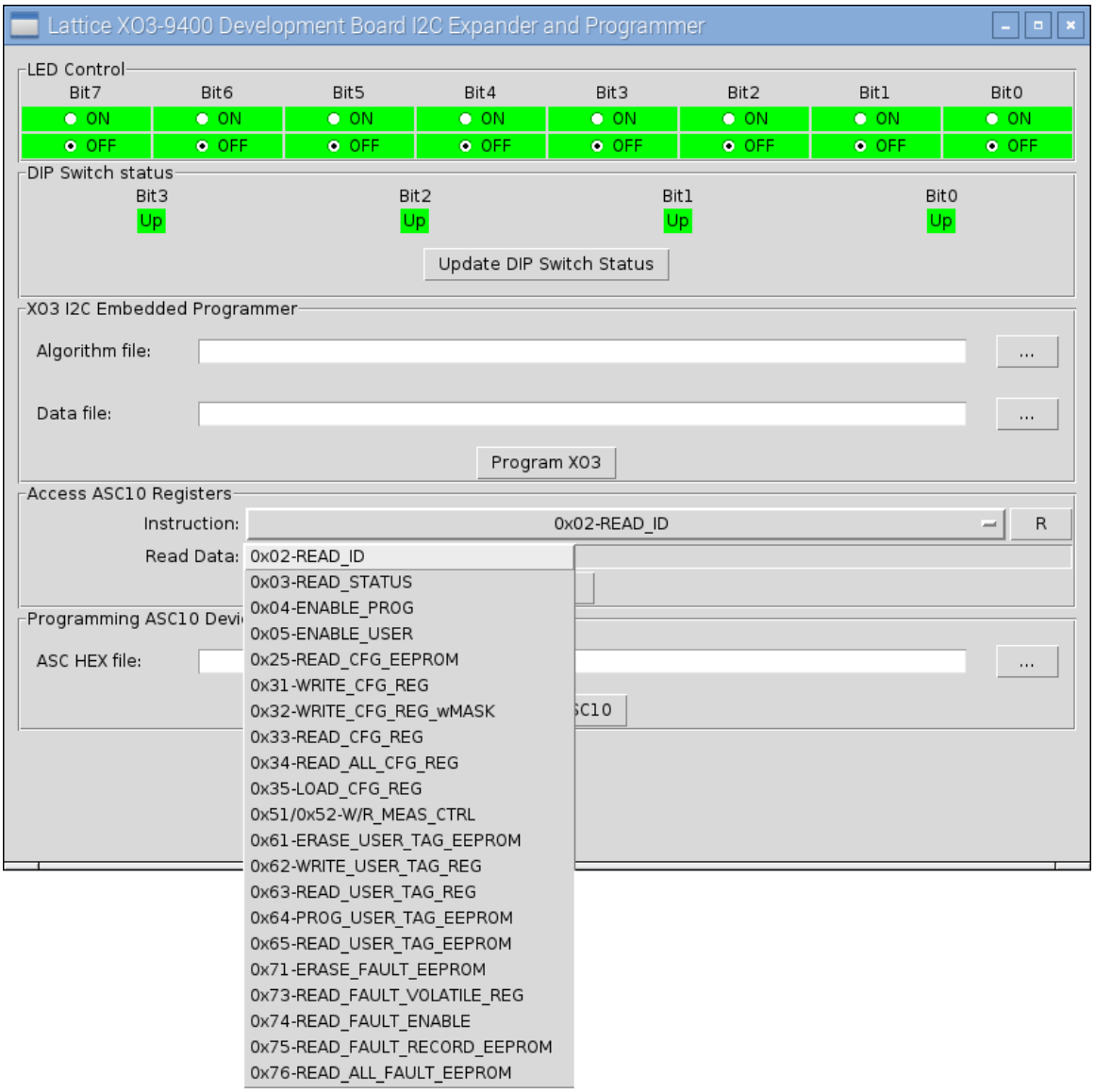


**Figure 11.1. Supported ASC10 I²C Commands**

To program L-ASC10 over I²C with the Raspberry Pi:

1. Make sure JP1 (Figure 3.1) is set to provide power for the L-ASC10 device. Or, the I²C Expander and Programmer will report I²C Write/Read Error.
2. To program the L-ASC10 device, use the **…** button in the ASC HEX file field to specify an ASC HEX file (Figure 11.2).

3. Browse to folder /home/pi/raspi/ispvm_embedded/i2c_embedded and select hardware_management_demo_ptm_ASC0.hex. Click **Open** (Figure 11.2).

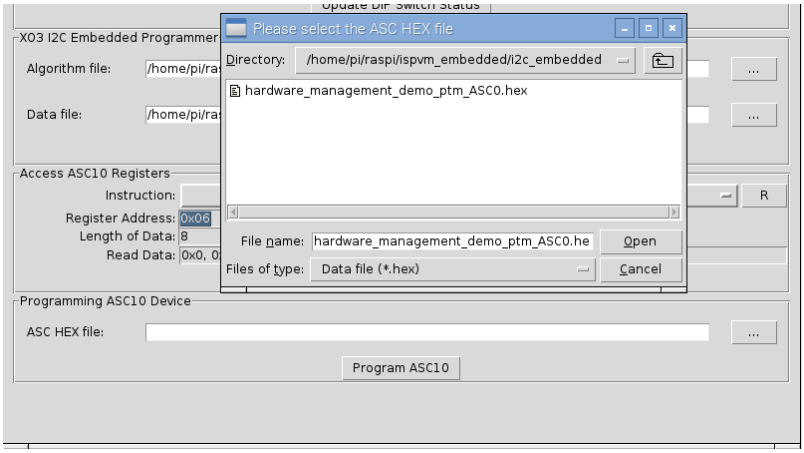


**Figure 11.2. Select File for ASC10 Device Program**

4. Click **Program ASC10** (Figure 11.2). The tool starts by checking the device ID, erasing and programming the device and verifying the programmed data (Figure 11.3).

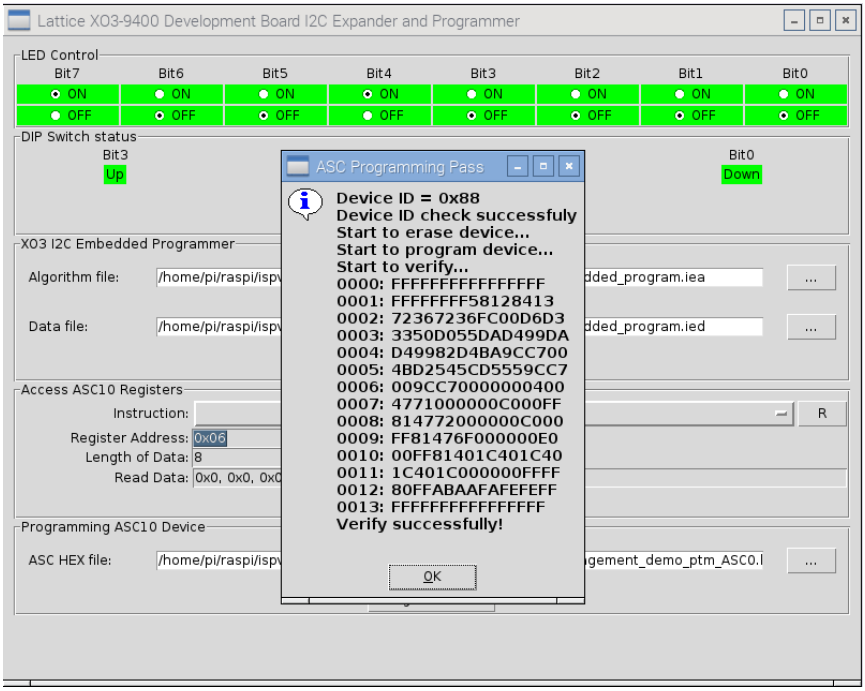


**Figure 11.3. Programming the L-ASC10 Device**

# 12. Update Reference Design for Customer Application

You can update and change the reference design for any application following steps below:

1. On your PC, start Lattice Diamond software and open the project at <*drive:*>/…/Raspi_I2C_expander/hardware/implementation directory.

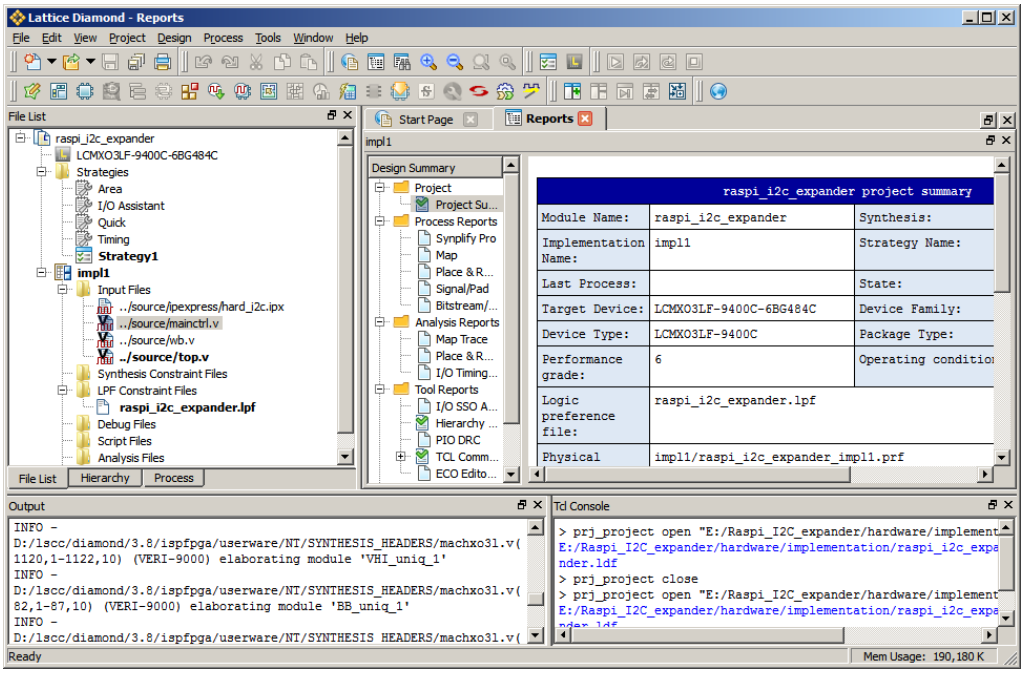


**Figure 12.1. Starting Diamond Software**

2. Double click to open the source file <*drive:*>/../source/mainctrl.v in the Source Editor (Figure 12.1).
3. Add new input or output ports to be expanded as the existing led[7:0] port and dipsw[3:0] port to the mainctrl.v file (Figure 12.2).
4. Update the control logic accordingly in the mainctrl.v.
5. Save your changes.

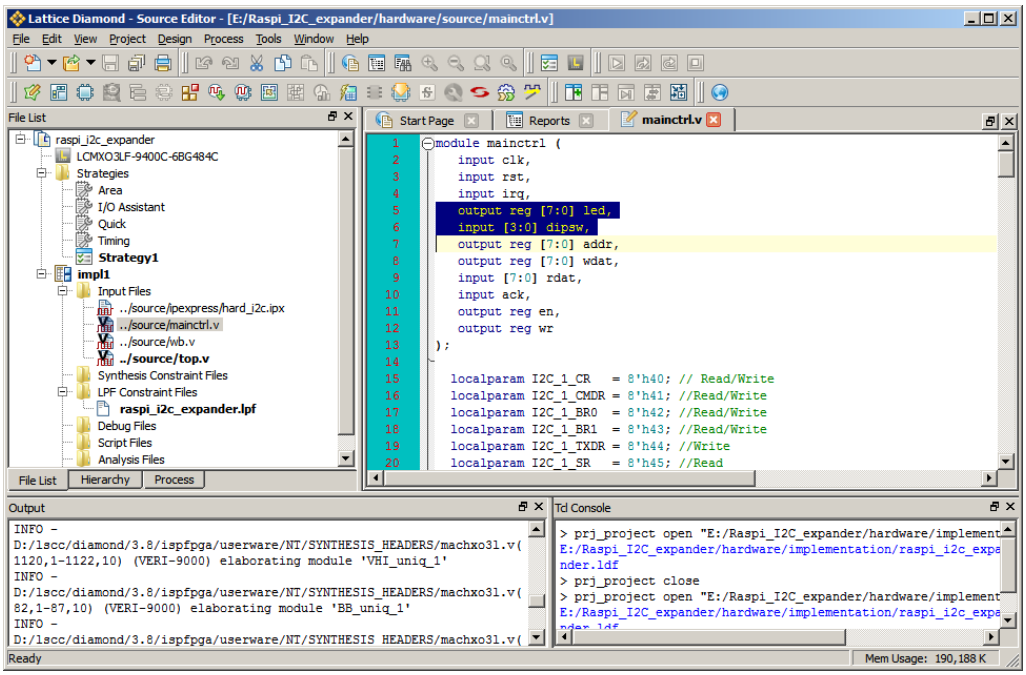


**Figure 12.2. Updating RTL**

6. Open Spreadsheet View (Figure 12.3) and make sure the I2C_PORT option is enabled.

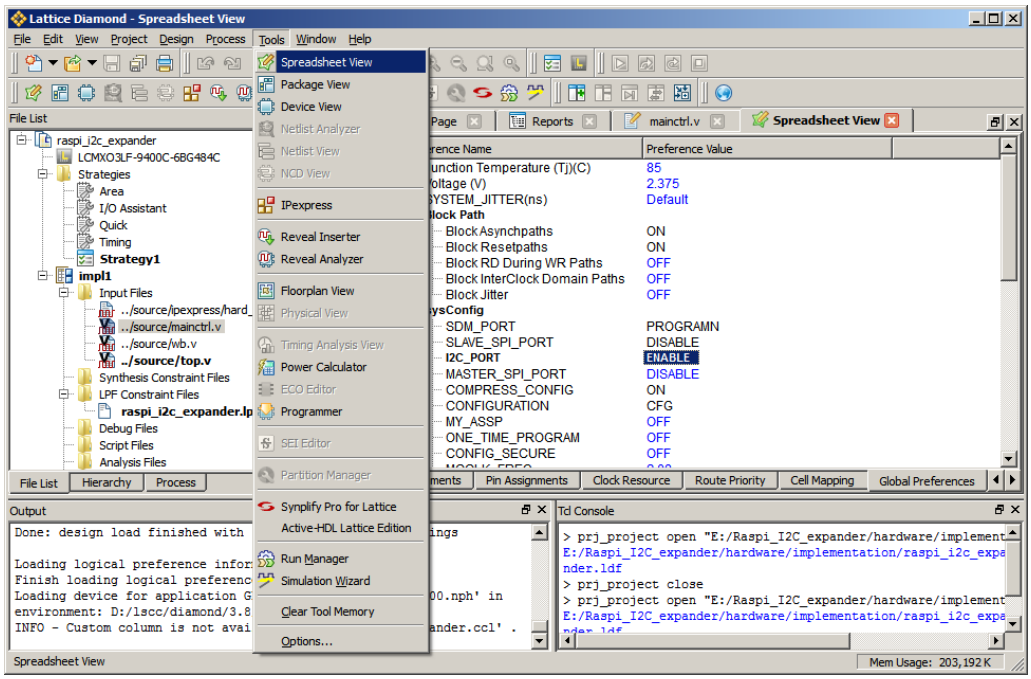


**Figure 12.3. Checking Global Preferences Settings**

7. Change to choose to open the Process tab from the left-side of the Lattice Diamond GUI. Double click the **Map Design** process (Figure 12.4). Assign pins for the newly-added ports to VERSA connectors reserved for users on the board. Save the changes.

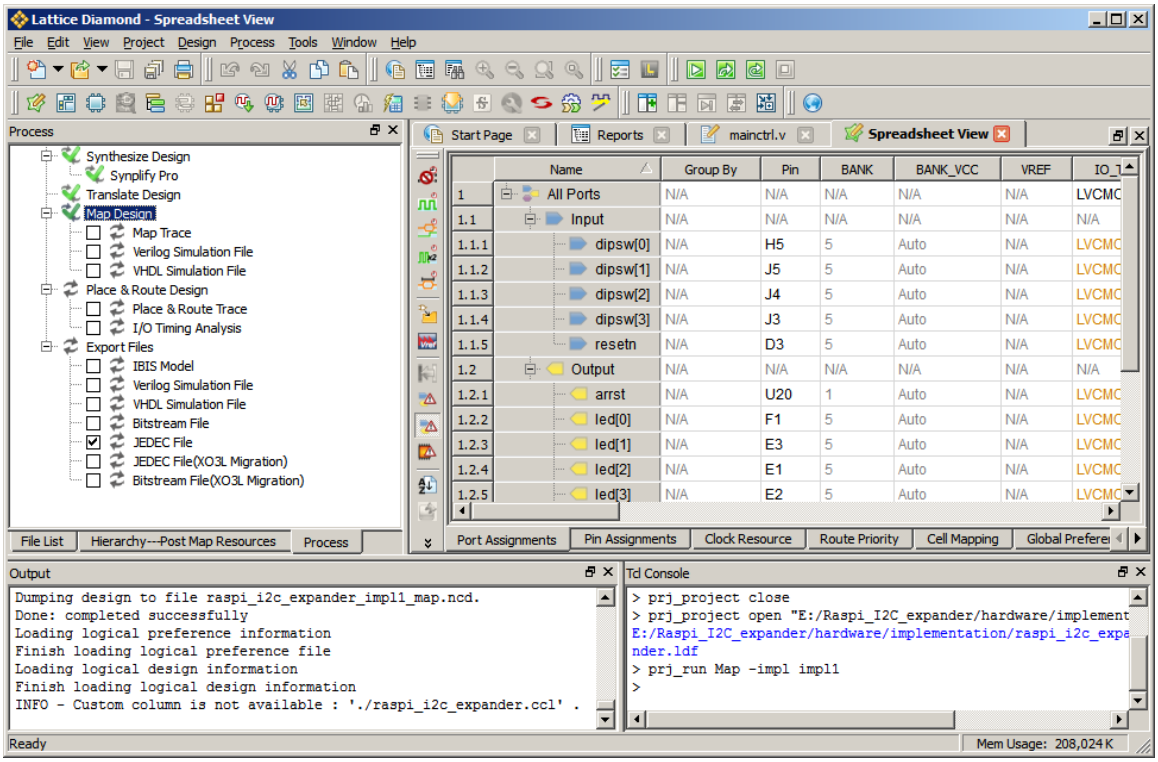


**Figure 12.4. Assigning the Newly-added Ports to VERSA Connectors**

8. Select the JEDEC File checkbox (Figure 12.5). Right click **JEDEC File** and select **Rerun All** to generate the new JEDEC file for programming the device.

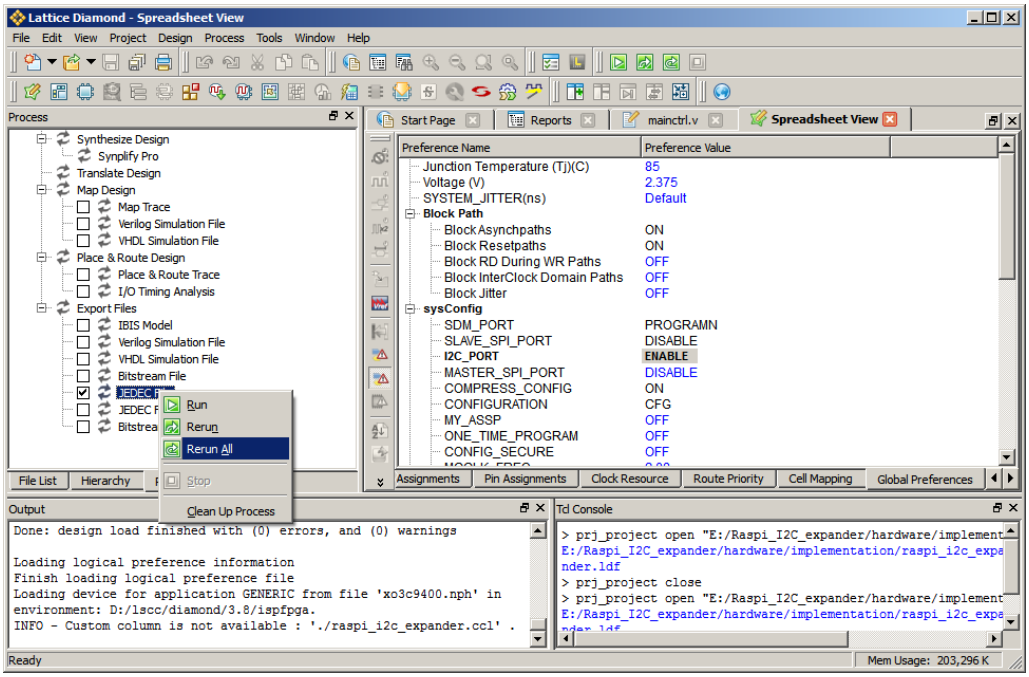


**Figure 12.5. Generating the JEDEC File**

9. Update the Python script file on Raspberry Pi based on the RTL changes (Figure 12.6).

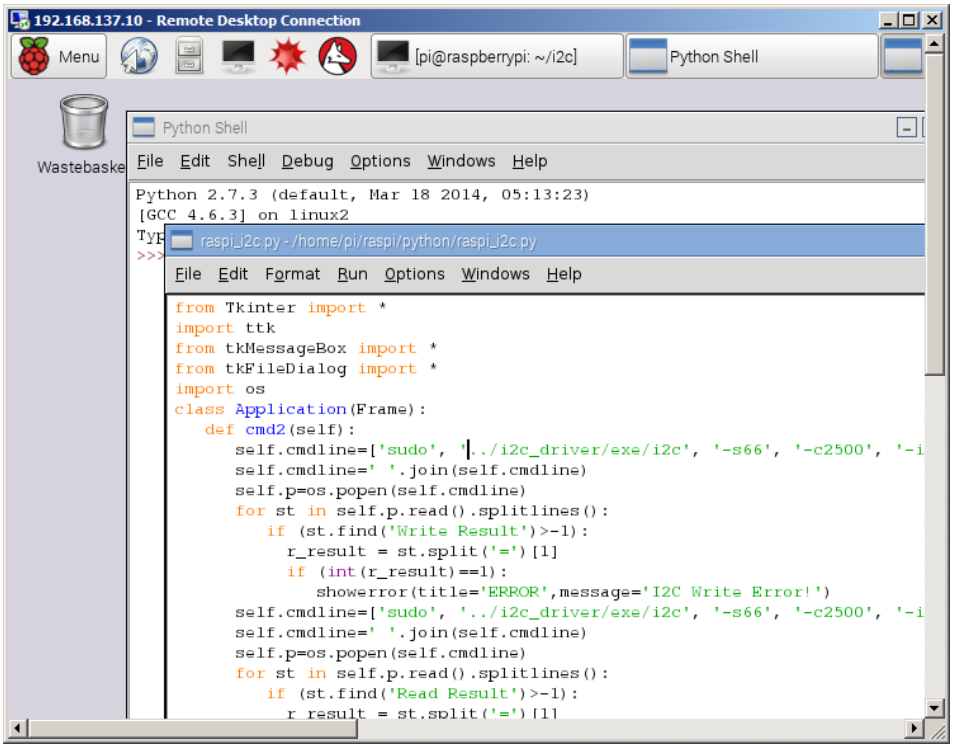


**Figure 12.6. Updating the Software on Raspberry Pi**

10. Re-run I²C Expander and Programmer as described in the Running the I2C Expander Demo on Raspberry Pi section. Now you can go back to the MachXO3-9400 Development Board to drive the input signals or to check the output signals on the VERSA connectors. They should change with the software operation. If any issue is found, you can go back to Step 3 updating your design till the newly-added ports function the same as expected.

# References

- DS1047, MachXO3 Family Data Sheet
- DS1042, L-ASC10 In-System Programmable Hardware Management Expander Data Sheet
- FPGA-EB-02004, MachXO3-9400 Development Board User Guide
- TN1279, MachXO3 Programming and Configuration Usage Guide

# Technical Support

For assistance, submit a technical support case at www.latticesemi.com/techsupport.

# Revision History

**Revision A, November 2017**

Initial production release.

7th Floor, 111 SW 5th Avenue
Portland, OR 97204, USA
T 503.268.8000
www.latticesemi.com