MachXO3D Hitless I/O and Hitless EBR Demo

User Guide

FPGA-UG-02069-1.1

November 2019

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
DIP	Dual In-line package
EBR	Embedded Block RAM
GSR	Global Set/Reset
MCB	Message Control Block
RAM	Random Access Memory
SRAM	Static Random-Access Memory

1. Introduction

In mission critical systems such as data centers, storage and networking, feature improvements and bug fixes are performed through background updates. Designers, however, avoid updating the control programmable logic device (PLD) because it forces a power cycle or reset of the entire system to enable the new algorithm to take effect. The PLD typically performs power management, reset management, glue logic, and other housekeeping functions on the board that should not be interrupted.

The Lattice Semiconductor MachXO3D™ control PLD offers designers the option of performing updates at the background and enabling new algorithms to take effect without interrupting board-level operations by eliminating the need to power-cycle or reset the device. The MachXO3D™ Hitless I/O and Hitless EBR Demo showcases this feature known as hitless or zero-downtime system update. The Lattice Diamond® software is used in this demo, enabling key features such as TransFR and Leave Alone.

This demo user guide describes the MachXO3D hitless I/O and hitless EBR technology and provides the details of the demo.

1.1. Demo Overview

System-critical functions are controlled and supervised by CPLD or FPGA logic. It can be time consuming, inconvenient and costly to remove these systems from service to perform updates to the reprogrammable logic. It is advantageous to update system control logic and state machines at the background without impacting dependent downstream circuits such as power supplies or alarms. The MachXO3D hitless I/O and hitless EBR feature is specifically designed to hold critical signal output states stable and keep memory context while the underlying logic is updated. The output states that are put on hold do not glitch during the update process, and the context in EBR is not lost or changed. These can optionally be used to preset the logic state machines to resume control at any desired operating point upon resumption of normal operation.

The hitless I/O and hitless EBR design uses silicon features built into the MachXO3D FPGA in concert with minimal additions to the user design logic. MachXO3D supports background reprogramming and TransFR mode reconfiguration. For hitless I/O, a simple multiplexer latch circuit is added to the critical user outputs and a simple common Message Control Block (MCB) is used for interfacing with the System Update Controller. The System Update Controller communicates with the MCB over a suitable channel, for example signal wire or I^2C, to coordinate the update event.

The hitless I/O and hitless EBR demo consists of five parts:

- Normal operation – Loading the original design. The output LEDs and D20 are under the original design control.
- Updating the flash image – Programming the new design into configuration flash at the background.
- Preparing for update – Communicating to the device regarding a design update using an external switch. The binary counter value is frozen. This step is only for hitless I/O.
- Updating the design – Reconfiguring the configuration SRAM with the new bitstream image contained in the configuration flash array using the TransFR and Leave Alone features.
- Resuming normal operation – The updated new design is active with the outputs driven by the updated design.

1.2. MachXO3D Development Board and Resources

The onboard buttons, DIP switches, segment LEDs such as D20, and LEDs of the MachXO3D development board are used to demonstrate the hitless I/O and hitless EBR feature of the MachXO3D device. Figure 1.1 shows the top side of the MachXO3D development board and resources used for the demo.

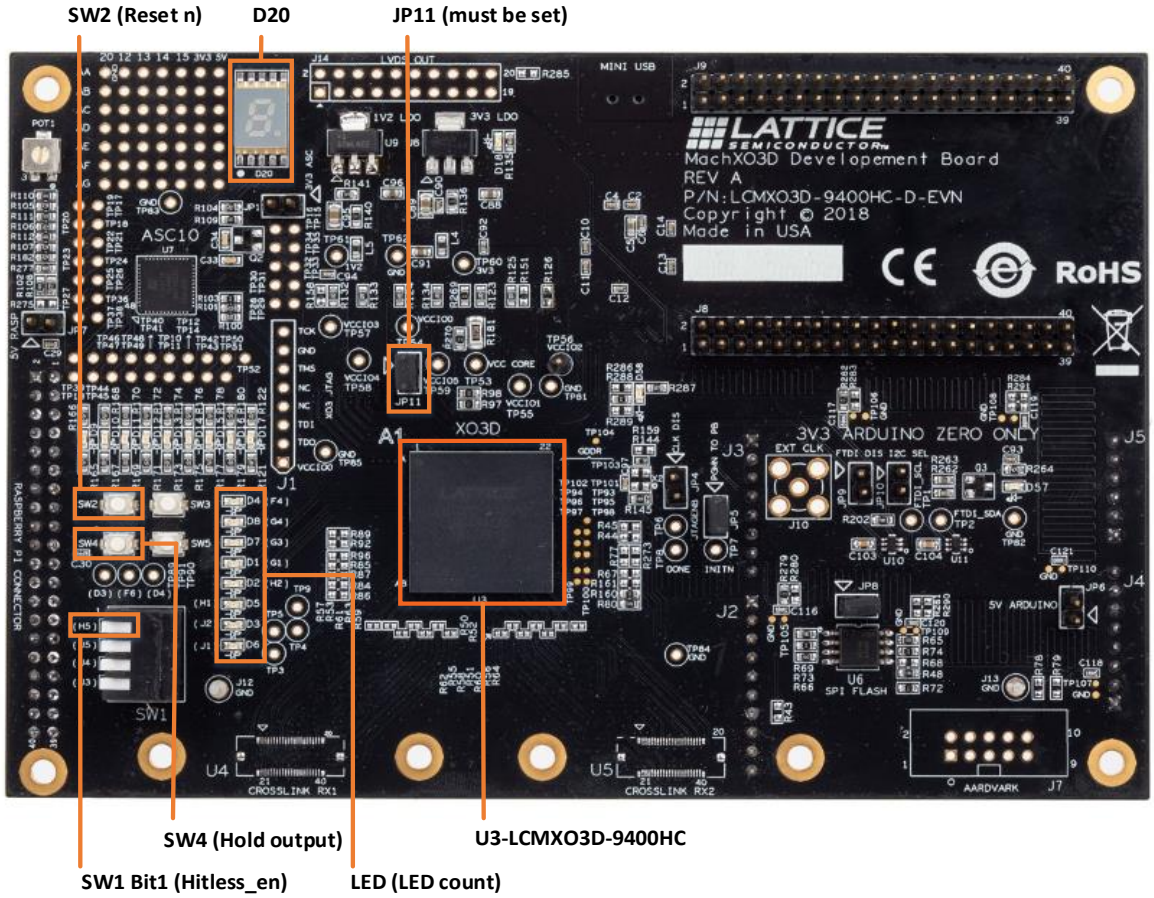

Figure 1.1 Demo Resources

2. Function Description

This section gives an introduction on hitless I/O and hitless EBR functions.

2.1. Hitless I/O

2.1.1. General Introduction

A Normal_operation signal is used to support the hitless I/O circuit. Normal_operation must have a default global reset signal GSR value of False. Following the release of the GSR, Normal_operation is typically asserted by the user logic to indicate that the user design circuit is free to operate. Normal_operation typically remains asserted until the start of the hitless I/O operation. A message from an external controller is used to deassert Normal_operation, typically after the new bitstream is programmed into flash memory and just prior to a sysConfig REFRESH command or PROGRAMN pin toggle. Figure 2.1 shows the hitless I/O design block diagram.

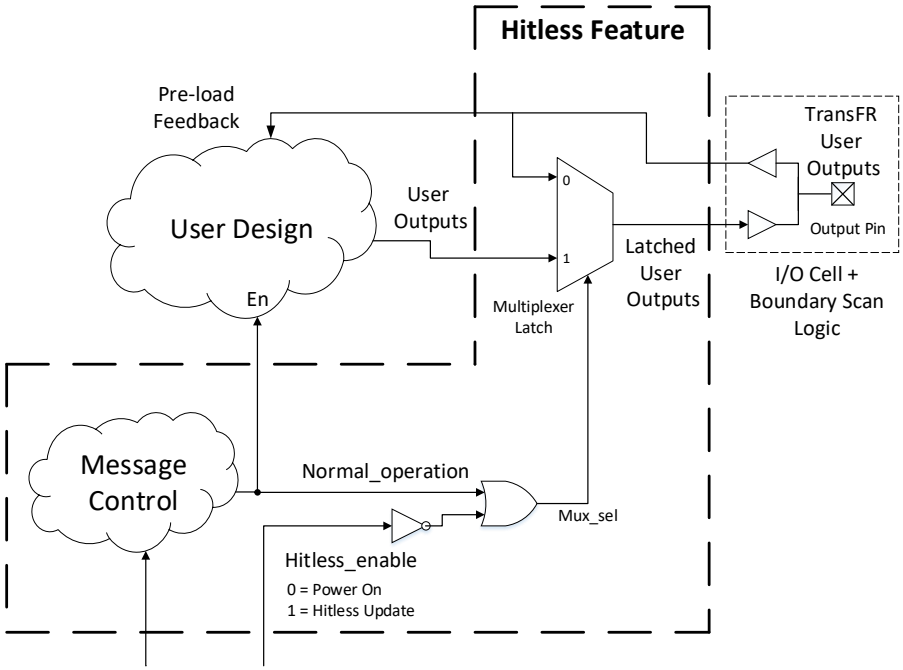

Figure 2.1. Hitless I/O Design Block Diagram

The user design outputs are captured and held with the soft 2:1 multiplexer latch. Normal_operation then remains deasserted until the MachXO3D device begins reconfiguration. As reconfiguration begins, the TransFR circuit in the boundary scan logic in the I/O cell latches the output value, Phase 2 of the Non-JTAG mode TransFR Sequence. Refer to Minimizing System Interruption During Configuration Using TransFR Technology (TN1087) for more details about TransFR. The TransFR boundary scan logic holds the output states through Phases 3 and 4 until the device wake-up is complete and the device re-enters user mode. The re-established soft 2:1 multiplexer re-acquires the TransFR cell value during the device wake-up sequence, Phase 4, and holds it until the assertion of Normal_operation.

Figure 2.2 shows the timing diagram of the hitless I/O demo design. For more details about timing, refer to Minimizing System Interruption During Configuration Using TransFR Technology (TN1087).

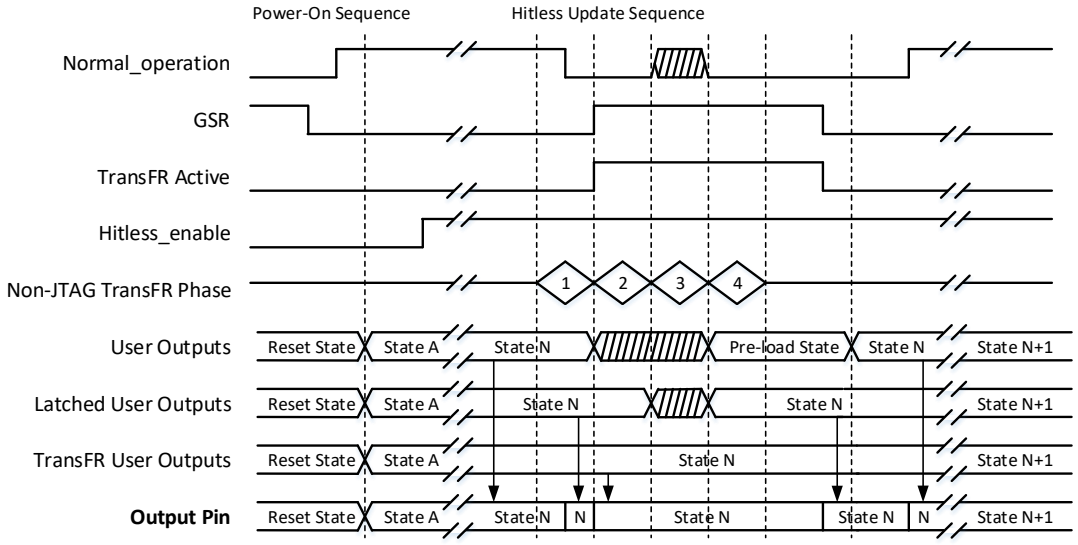

Figure 2.2. Hitless I/O Demo Timing Diagram

The hitless I/O logic must cover two primary scenarios when the FPGA enters user mode: power-up and hitless reconfiguration. The control input Hitless_en is driven by an external controller to determine which scenario is active. If Hitless_en is false, the user logic starts from a power-up reset state. If true, the user logic can utilize the values held in the TransFR boundary scan I/O cells to initialize the control logic to resume from the pre-update state. See Table 2.1 for the truth table showing both the general behavior of the feature and the specific behavior of the demonstration design.

Table 2.1. Truth Table for Normal_Operation and Hitless_Enable

Hitless_enable	Normal_operation	User Design Output	Demo Design Output
X	1	Normal Operation	Normal Counter Operation
0	0	User Defined Power-up State	0 (LED's ON)*
1	0	User Defined Hold/Preload State	Feedback Value

***Note**: All LED cathodes are driven by the design output. As such, LED ON represents 0 and LED OFF represents 1.

As shown in Figure 2.3, the hitless I/O design is divided into three major blocks:

- User design block
- Multiplexer latch to hold outputs
- Message control block

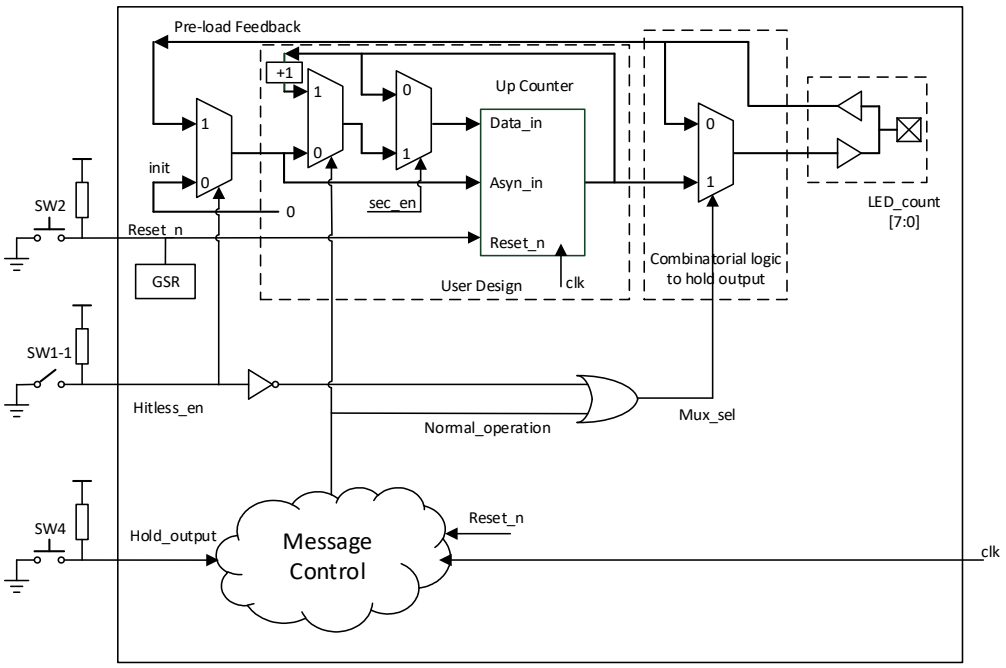

Figure 2.3. TransFR I/O Cell Simplified Block Diagram

For detailed pin/port descriptions, see the Port Assignments and Descriptions section.

2.1.2. User Design Block

The user design block contains the original user logic. In this case, it is a simple 8-bit up-counter design. The original user logic is updated to include two accommodations that support the hitless I/O process:

- Enable input
- Addition of a multiplexer latch to select the asynchronous reset Preset/Restore value.

In the demonstration, this block is updated to become a down-counter. The reset_n logic is placed on the GSR net by the software.

2.1.3. Multiplexer Latch to Hold Outputs

The hitless I/O design also features multiplexer latch(es) and a few basic logic cells added to the top level to hold the outputs stable. A 2:1 multiplexer latch is instantiated for each original output signal which is to become hitless, and the output is changed into a bidirectional I/O. A common combinatorial logic path is instantiated for the Hitless_en control signal. The path must be 100% combinatorial so that the 2:1 muxes are controlled by the state of Hitless_en prior to the release of GSR.

2.1.4. Message Control Block

This hitless I/O design addition is used to communicate messages to the hitless top level design that the TransFR operation is about to occur. SW4 represents the external controller as described in the General Introduction section. The message is conveyed by closing SW4, a momentary switch on the MachXO3D development board. The closing edge, also noted as falling edge, of Hold_output is detected and used to negate Normal_operation until the device is refreshed.

Following the FPGA image update, Normal_operation is re-asserted after a small delay, providing time for the circuit to synchronize to the previous counter state as preserved in the LED_count I/O cells.

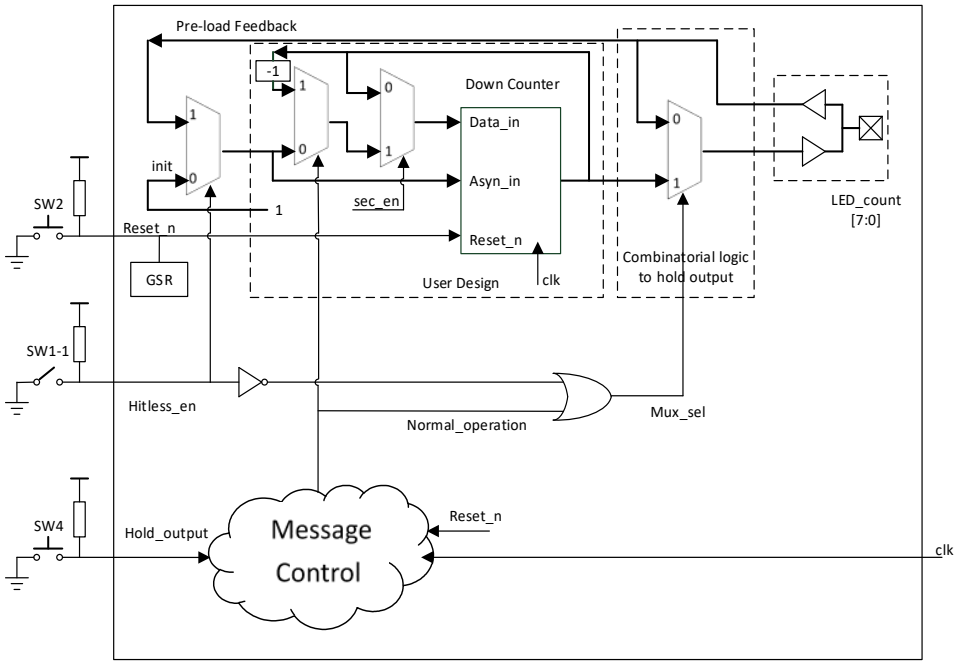

Figure 2.4. Hitless I/O New Block Diagram

A second, nearly identical design project is included as part of the hitless I/O and hitless EBR demonstration. As shown in Figure 2.4, the user design block is changed to include a down counter block in place of the prior up counter, and the EBR initializing context is changed to 4'h0, instead of the prior context which is from 4'h0 to 4'hF. All the other blocks remain untouched. You can switch between the designs without causing glitches or state changes to the LED outputs and EBR context.

2.2. Hitless EBR

Hitless EBR is part of the Demo, as shown in Figure 2.5, module rom_addr_gen is to generate the addresses of ROM16x8, and the generated addresses rom_addr is increased from 4'h0 to 4'hF every second. ROM16x8 is the initial value of ROM16x8 is a lookup table, which has four inputs, rom_addr, and eight outputs for D20 segment LED control. ROM16x8 is implemented with EBR, which initializes its context during power-cycle. To describe process that how the Hitless EBR works, two implementations, original_up_count_impl1 and new_down_cnt_impl2, are created in the demo project. In original_up_count_impl1, the initial value of ROM16x8 is for 0~9 and A~F display of D20 segment LED. And, in new_down_cnt_impl2, the initial value of ROM16x8 is all 0s. When the MachXO3D device is background reprogramming and TransFR mode reconfiguration, the EBR used in this demo keeps its context while the underlying logic is updated.

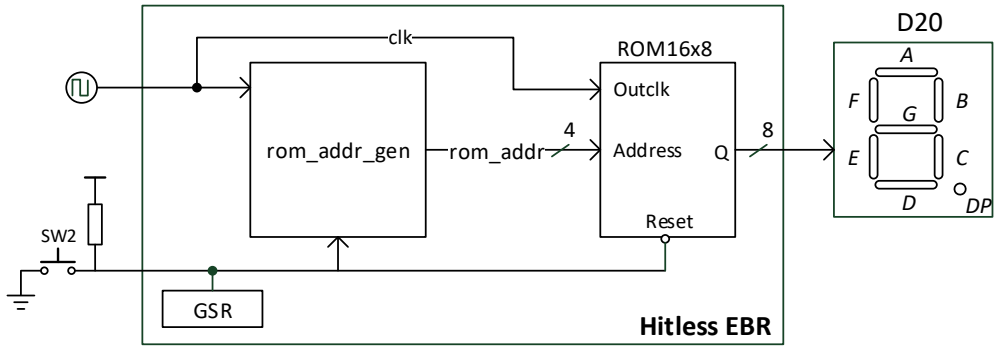

Figure 2.5. Hitless EBR Design Block Diagram

After programming MachXO3D_Hitless_IO_demo_original_up_count_impl1_a.jed file, which was generated from original_up_count_impl1 without TransFR mode reconfiguration, the LED count shifts from D4 to D6, and D20 also increases from 4'h0 to 4'hF by 1 every second, which is shown in Figure 2.6.

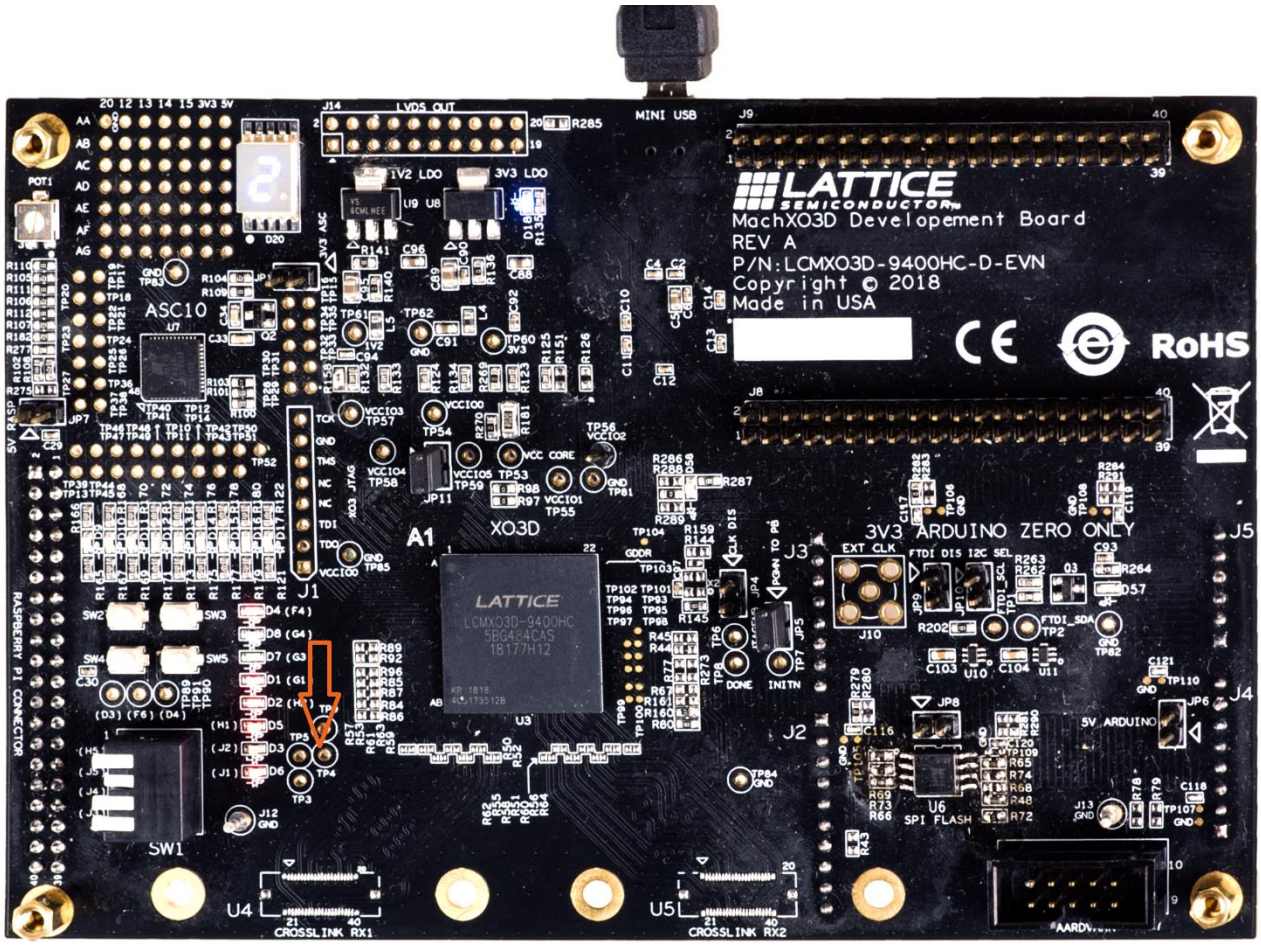

Figure 2.6. Original Implementation

After programming MachXO3D_Hitless_IO_demo_new_down_cnt_impl2_a.jed file, which was generated from new_down_cnt_impl2 without TransFR mode reconfiguration, the LED count shifts from D6 to D4, and D20 is frozen with '8.', which is shown in Figure 2.7.

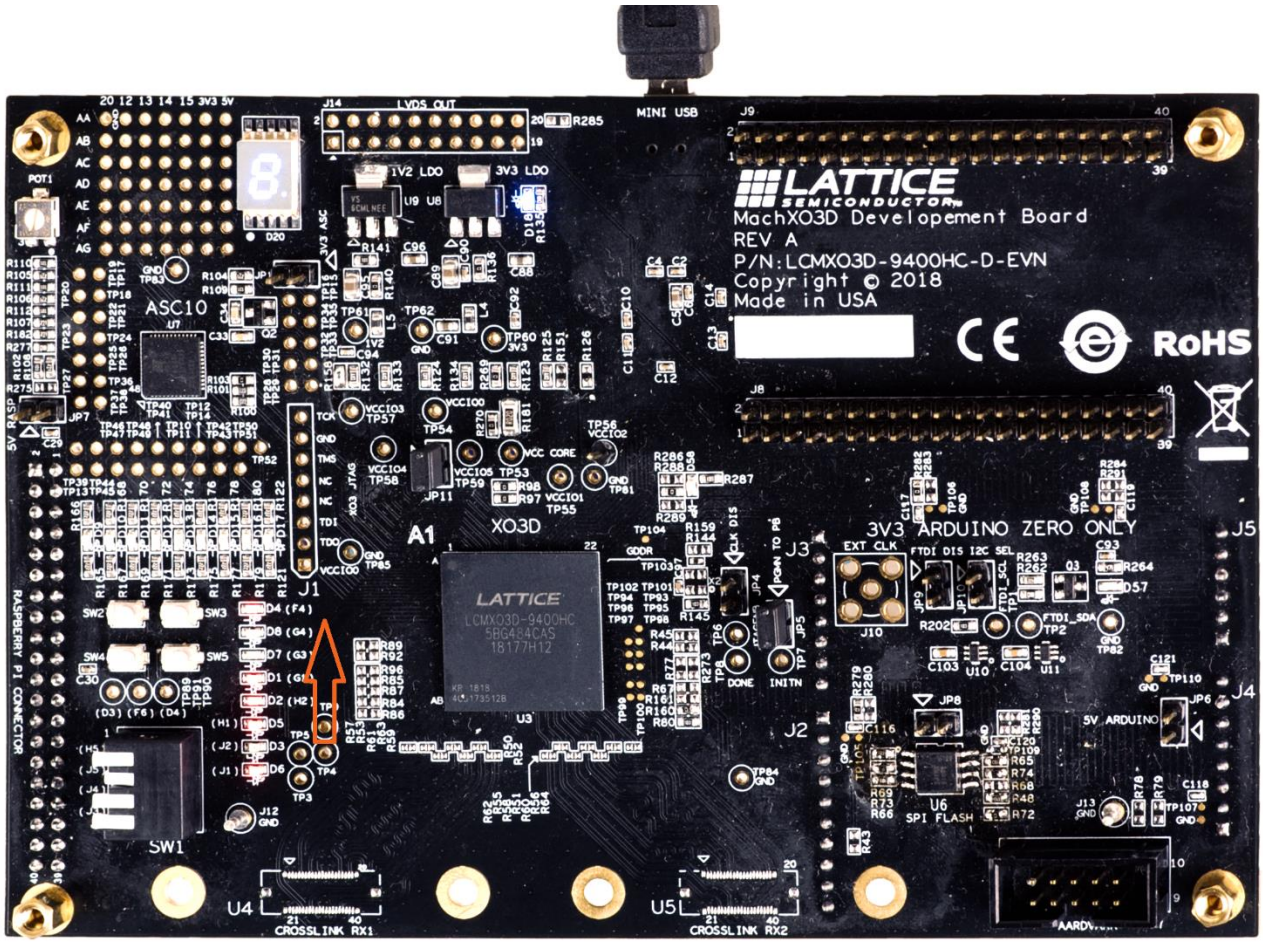

Figure 2.7. New Implementation

Therefore, you can know if the latest reconfiguration is active by LED count shifting direction, and verify whether or not EBR context update by D20 segment LED after TransFR reconfiguration.

3. Demo Package

The demo package includes the following:

- Lattice Diamond project and preference files
- JEDEC files (*.JED) for programming the MachXO3D internal configuration flash

3.1. Hardware Requirements

To run the demo, the following hardware are required:

- PC running Windows 7 operating system
- MachXO3D development board
- Mini USB cable for programming the MachXO3D device

3.2. Software Requirements

To run the demo, the following software are required:

- Lattice Diamond version 3.11 or later
- Lattice Diamond Programmer software for bitstream downloading

Note: These software programs are available at www.latticesemi.com/en/Products/DesignSoftwareAndIP.

4. Port Assignments and Descriptions

Table 4.1. FPGA Demo Design Ports

Port Name	I/O Type	Width	Description
Reset_n	Input	1	Asynchronous reset, active low. Button SW2 on the board.
Clk	Input	1	Input clock from crystal
Hitless_en	Input	1	Toggle switch SW1 BIT1 to differentiate between the power on and hitless I/O operation. Down (0) – Power on Up (1) – Hitless I/O
hold_output	Input	1	Push-button switch (SW4) to hold the output state.
LED_count	Output	8	LEDs indicating the output values.
SEG_LED_A	Output	1	A led of D20
SEG_LED_B	Output	1	B led of D20
SEG_LED_C	Output	1	C led of D20
SEG_LED_D	Output	1	D led of D20
SEG_LED_E	Output	1	E led of D20
SEG_LED_F	Output	1	F led of D20
SEG_LED_G	Output	1	G led of D20
SEG_LED_DP	Output	1	DP led of D20

5. Demo Package Directory Structure

Figure 5.1 shows the demo package directory structure.

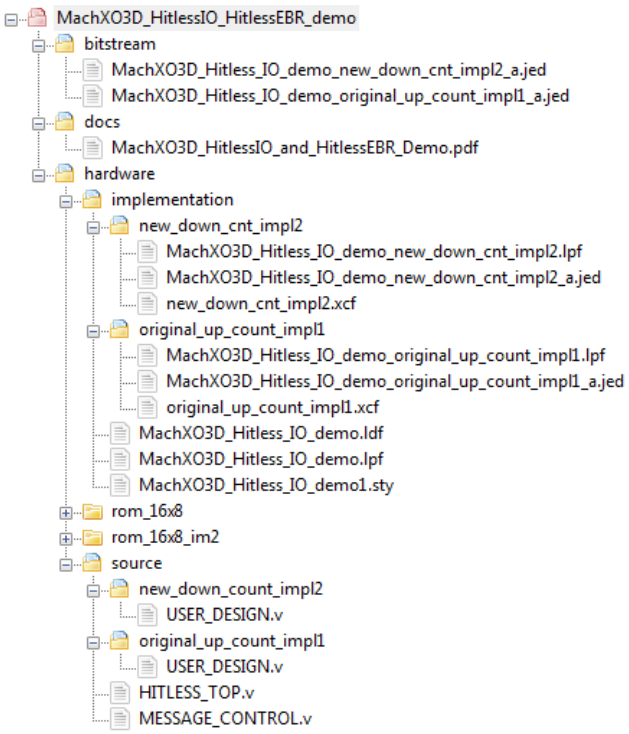

Figure 5.1. Demo Package Directory Structure

6. Running the Demo

6.1. Programming the Device

Load the initial design. The onboard output LEDs start counting upwards. To program the device:

1. Launch Diamond Programmer Version 3.11 or above. In the Getting Started dialog box (Figure 6.1), select **Create a new project file from JTAG scan**. Click **OK**.

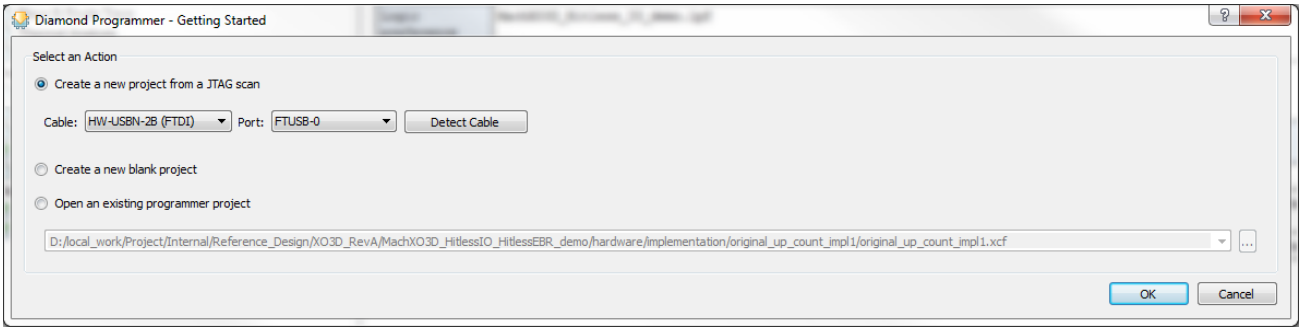

Figure 6.1. Diamond Programmer Getting Started Dialog

2. The Diamond Programmer starts scanning the board attached to the USB cable.

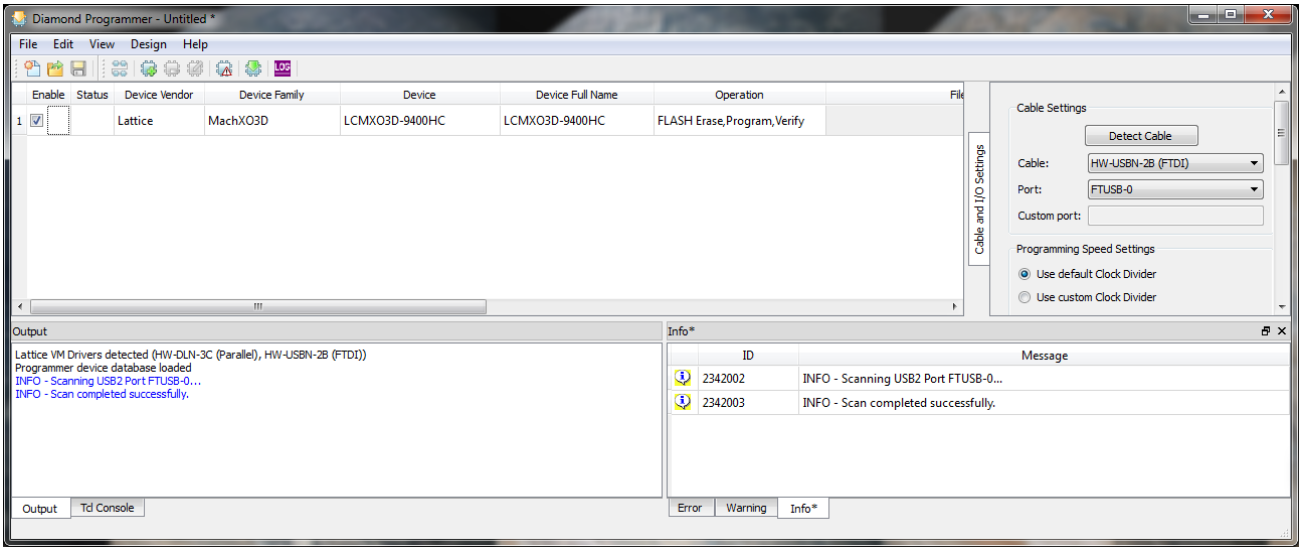

Figure 6.2. Diamond Programmer Main Interface

3. Click to highlight the device row. Select **Edit > Device Properties** from the menu bar (Figure 6.3). You can edit the access mode, operation mode, and select the programming file in this Device Properties dialog (Figure 6.4).

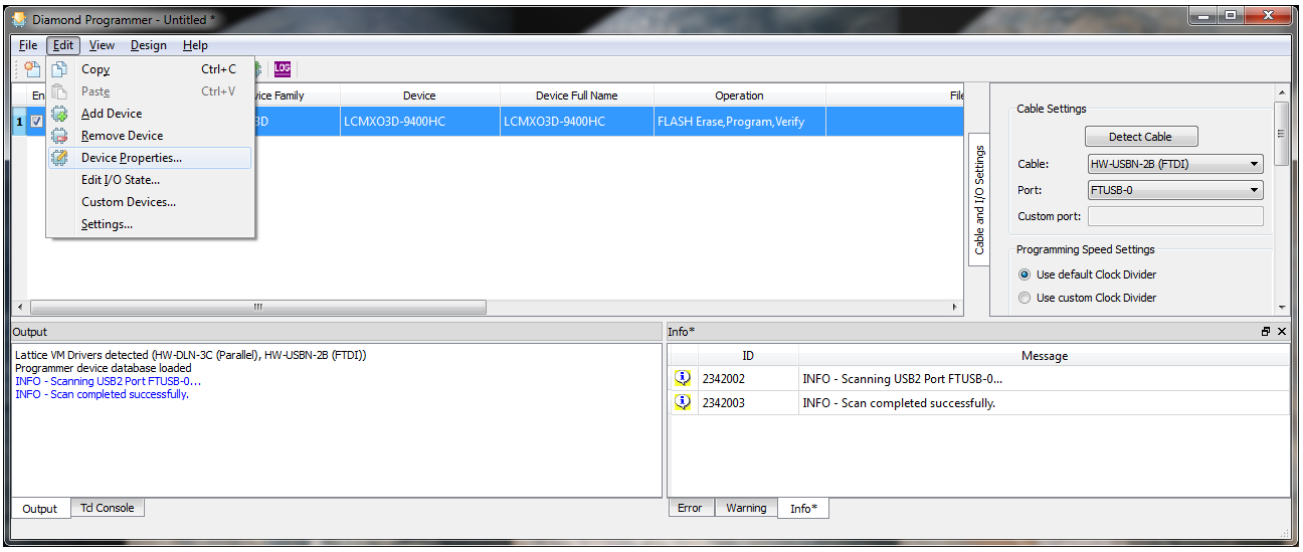

Figure 6.3. Edit > Device Properties Option

4. In the Device Properties dialog box (Figure 6.4), select **Flash Programming Mode** from the **Access mode** dropdown menu, and **Flash Erase, Program, Verify** from the **Operation** dropdown menu. Browse to select the bitstream file **.\bitstream\MachXO3D_Hitless_IO_demo_original_up_count_impl1.jed** in the **Programming file** field. Click **OK**.

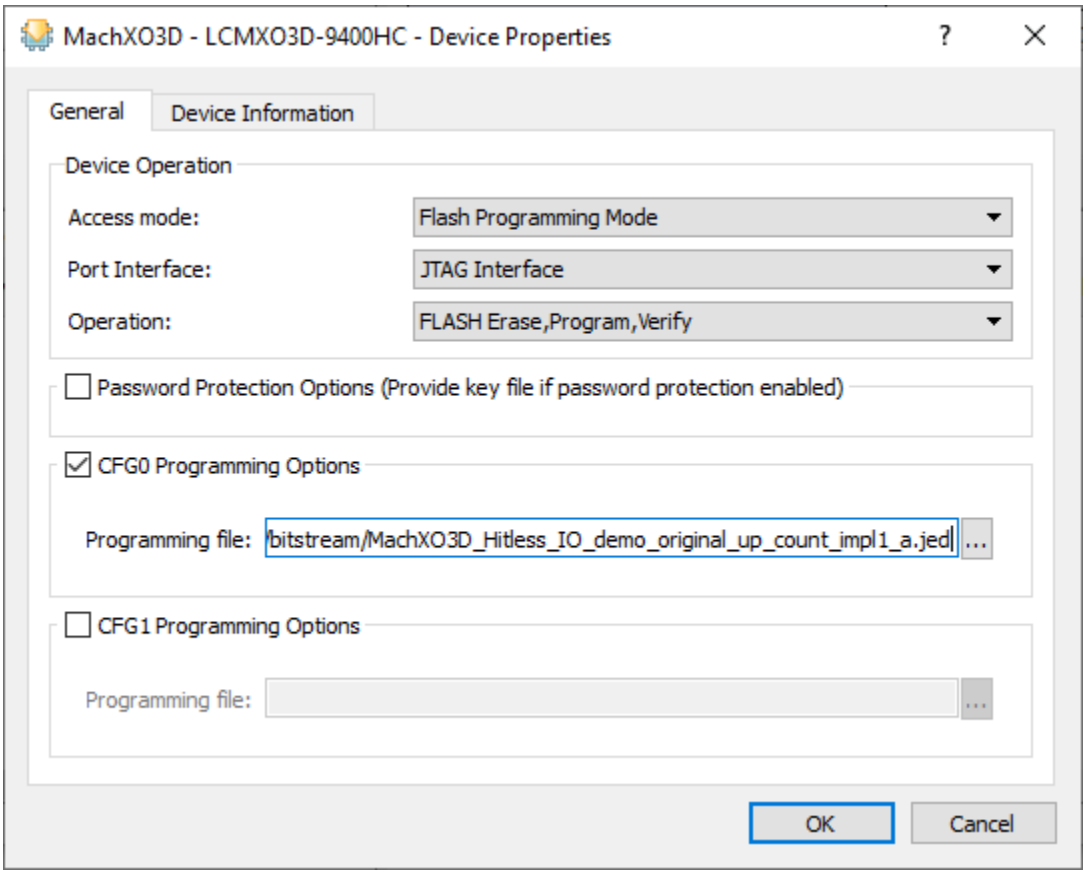

Figure 6.4. Browse Programming File

5. Now you can program the device. Select **Design > Program** (Figure 6.5), or click the **Program** button ( ) from the toolbar.

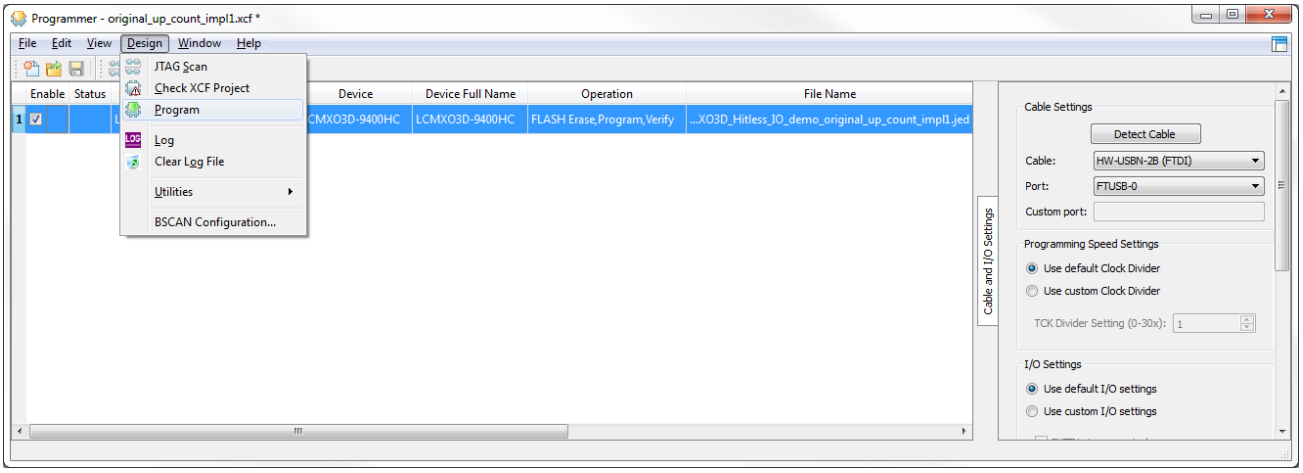

Figure 6.5. Programming the Device

When the programming of the device is completed, the LEDs on the MachXO3D development board start counting up from zero. LED OFF represents 1. LED ON represents 0. D20 increases from 4'h0 to 4'hF by 1 every second.

6.2. Updating the Flash Image

To update the flash image:

1. In Diamond Programmer, click to highlight the device row. From the menu bar, select **Edit > Device Properties** (Figure 6.6).

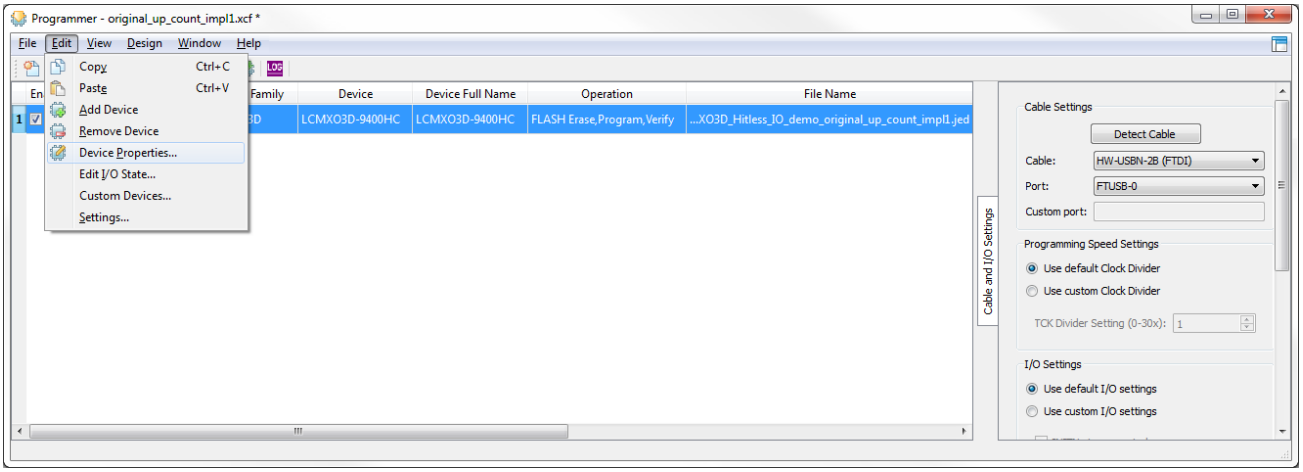

Figure 6.6. Device Properties Option

2. In the pop-up Device Properties dialog, select **Flash Background Mode** from the **Access mode** drop-down menu (Figure 6.7).

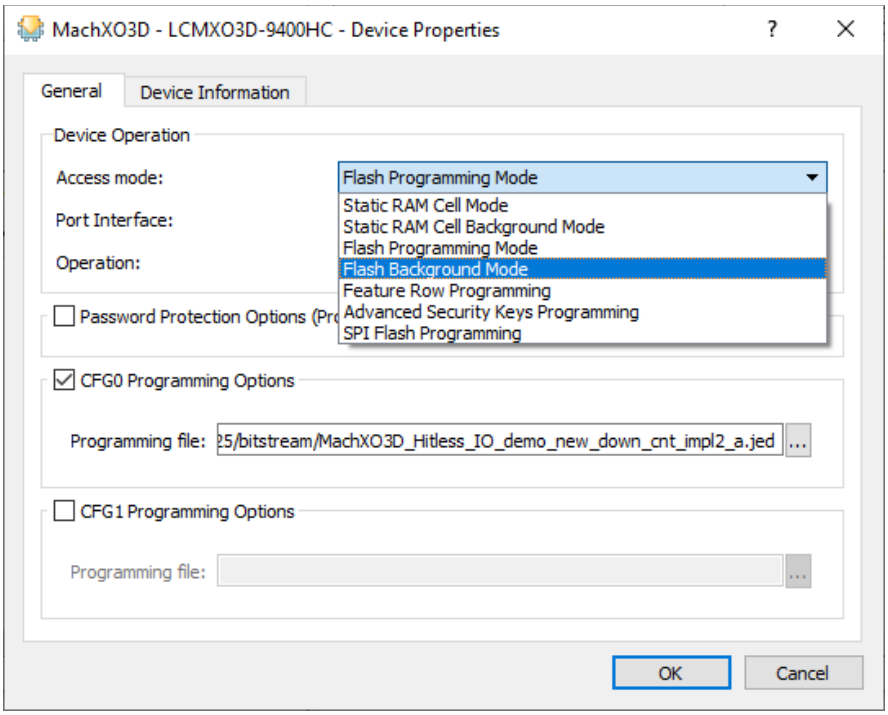

Figure 6.7. Access Mode Options

3. Select **XFLASH Erase, Program, Verify** from the **Operation** dropdown menu (Figure 6.8).

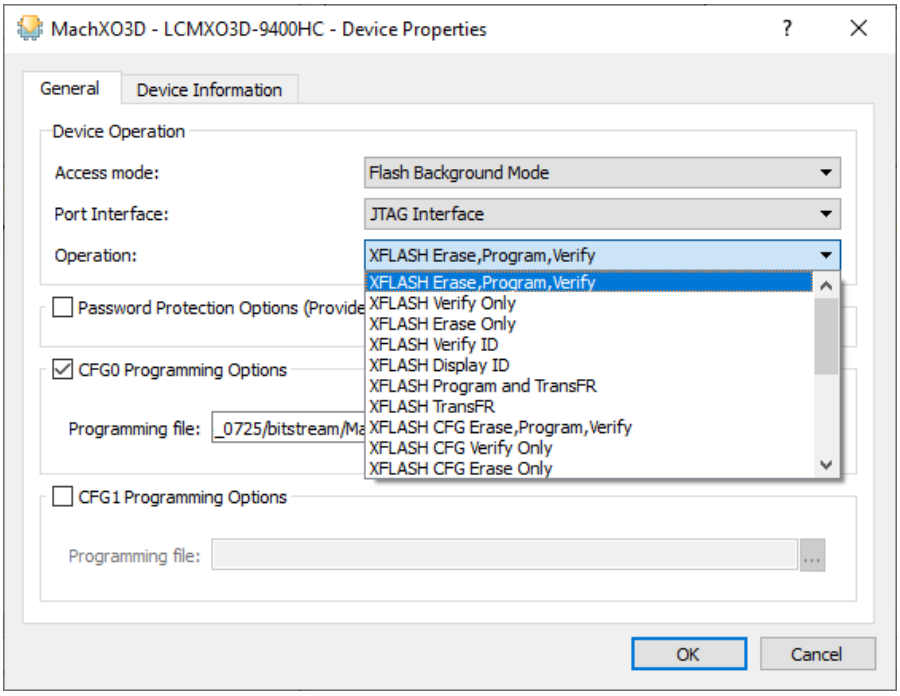

Figure 6.8. Operation Options

4. Select the bitstream file **.\bitstream\MachXO3D_Hitless_IO_demo_new_down_cnt_impl2.jed** in the **Programming file** field (Figure 6.9). Click **OK**.

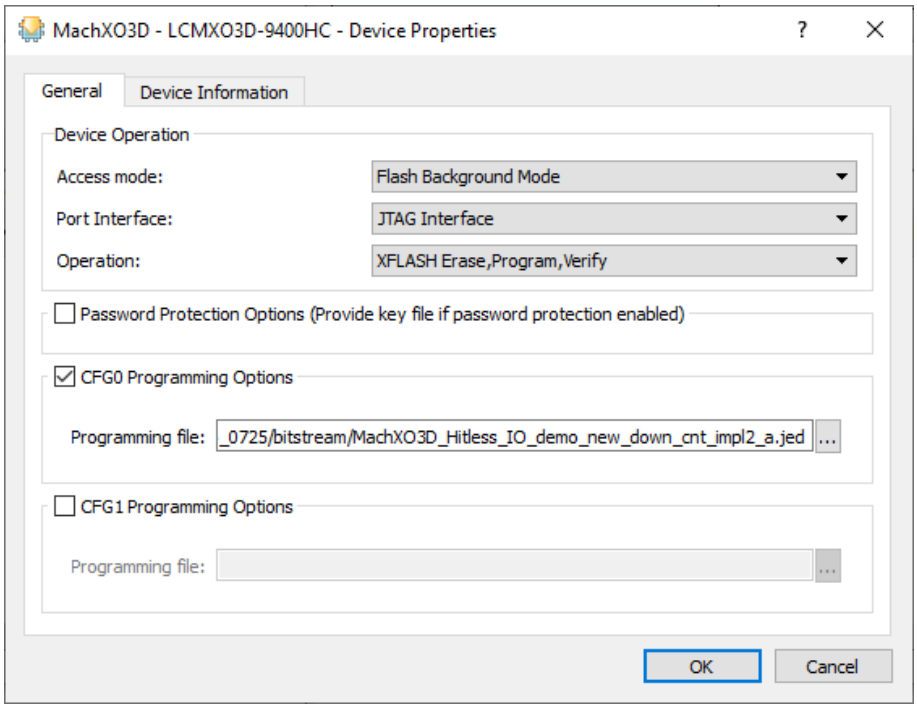

Figure 6.9. Browse Programming File

5. Now you can program the bitstream file to the device. Select **Design > Program** (Figure 6.10), or click the **Program** button ( ) from the toolbar.

During programming, the LEDs do not change counting state nor glitch. D20 continues to count up from 4'h0 to 4'hF by 1 every second. This can be confirmed by using an oscilloscope.

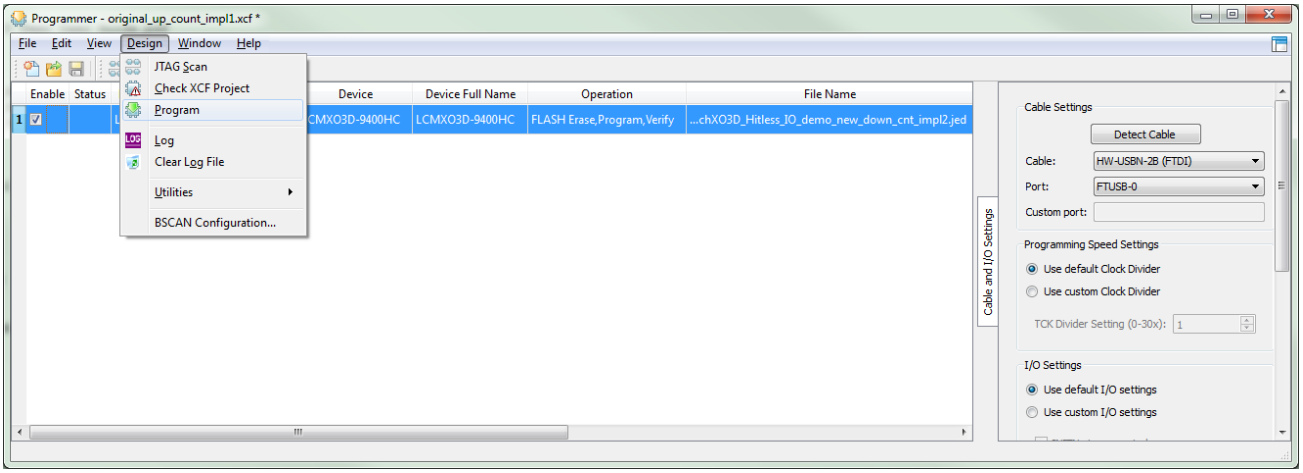

Figure 6.10. Programming the Device

6.3. Preparing for Update

Before the new design is transferred to active SRAM, it is necessary to communicate with the device regarding the imminent update.

In this demo, an external switch is used to communicate with the device regarding the design update. Check that all bits of SW1 are Up. Then press the button SW4 momentarily to freeze the LED binary count output on LED7~0 (Figure 6.11).

Note that if SW1 BIT1 is Down, the counter resets to the power up state when button SW4 is pressed. LED OFF represents 1. LED ON represents 0. When SW1 BIT1 is up, if SW4 is not pressed to freeze the outputs, the outputs may glitch during the update step. Besides, D20 keeps increasing by 1 every second, regardless of SW1 bit1 or SW4 is pressed or not. D20 is only driven by EBR context.

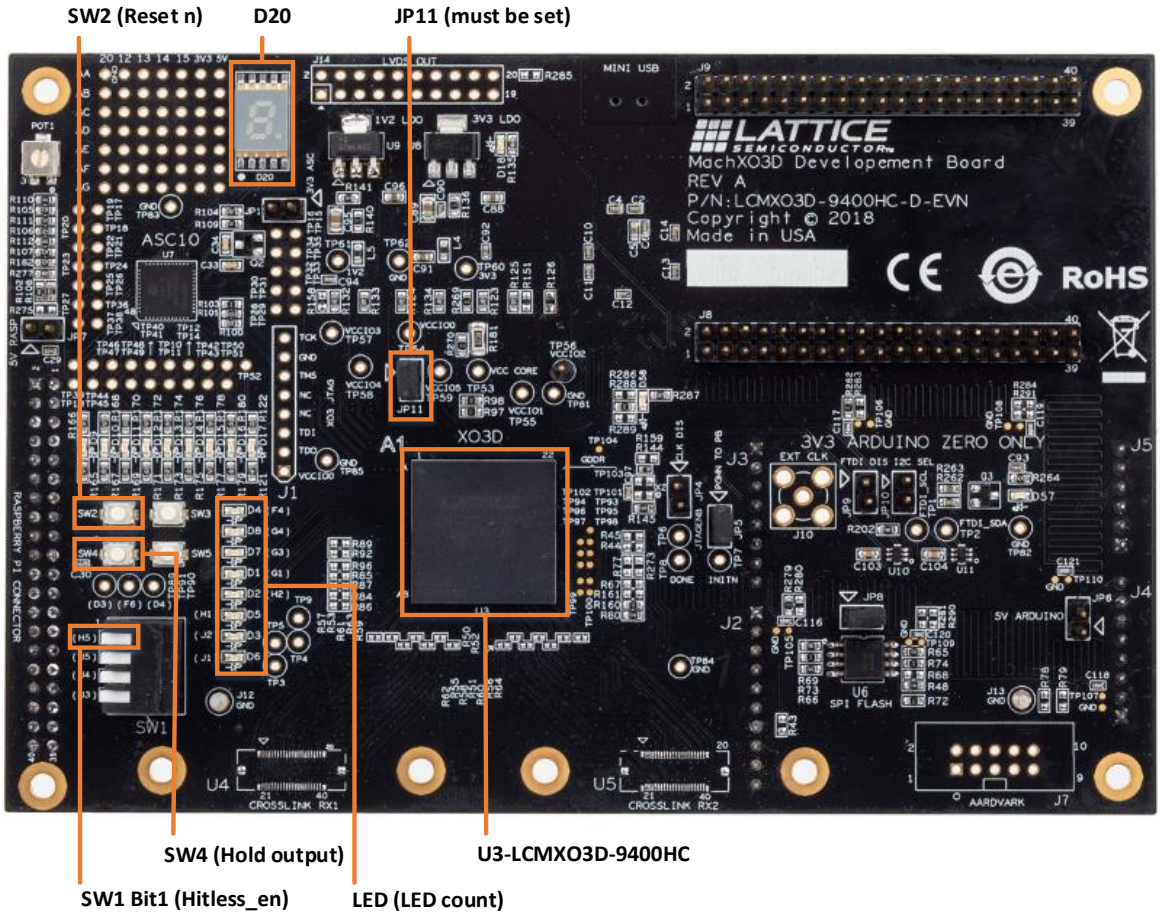

Figure 6.11. Preparations for Updating the Design

6.4. Updating the Design

To update the design:

1. In Diamond Programmer, highlight the device row by clicking anywhere of the row. Select **Edit** > **Edit I/O State** … (Figure 6.12).

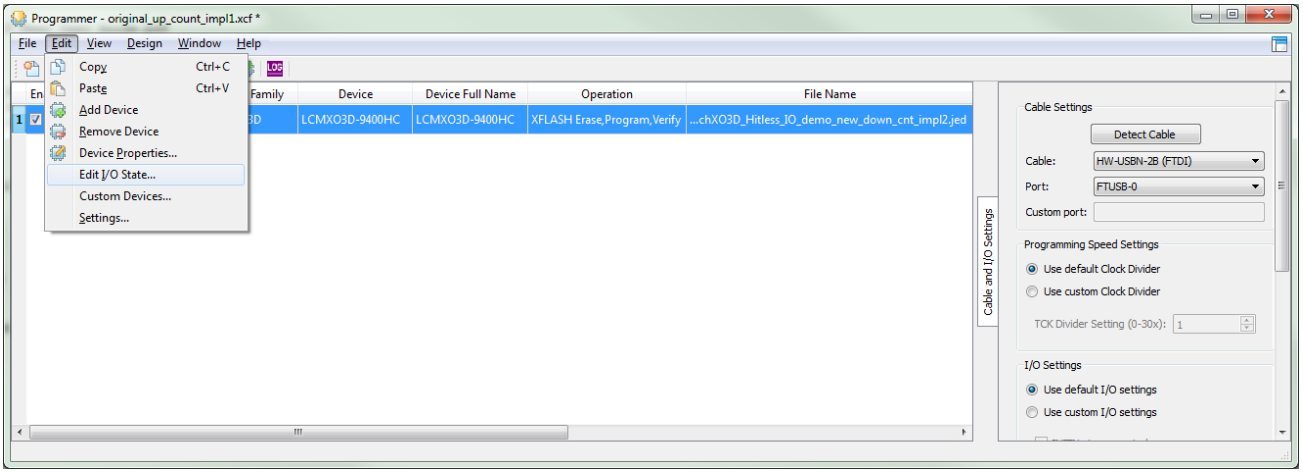

Figure 6.12. Edit I/O State Option

2. In the Edit I/O State dialog box, select **Leave Alone** from the **I/O state** dropdown menu (Figure 6.13). Click **OK** to confirm this selection.

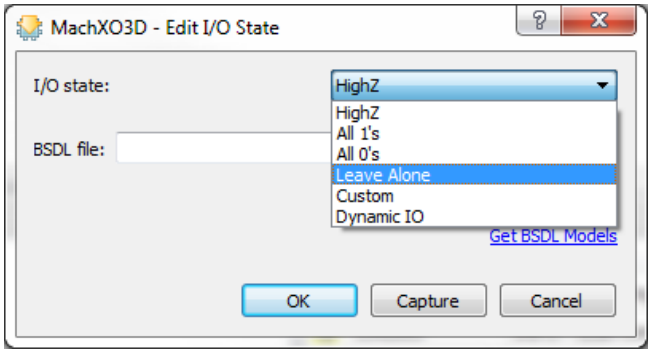

Figure 6.13. Edit I/O State Dialog Box

3. Click the device row to highlight it and from the menu bar, select **Edit > Device Properties**. This allows you to edit the access mode as well as operation.

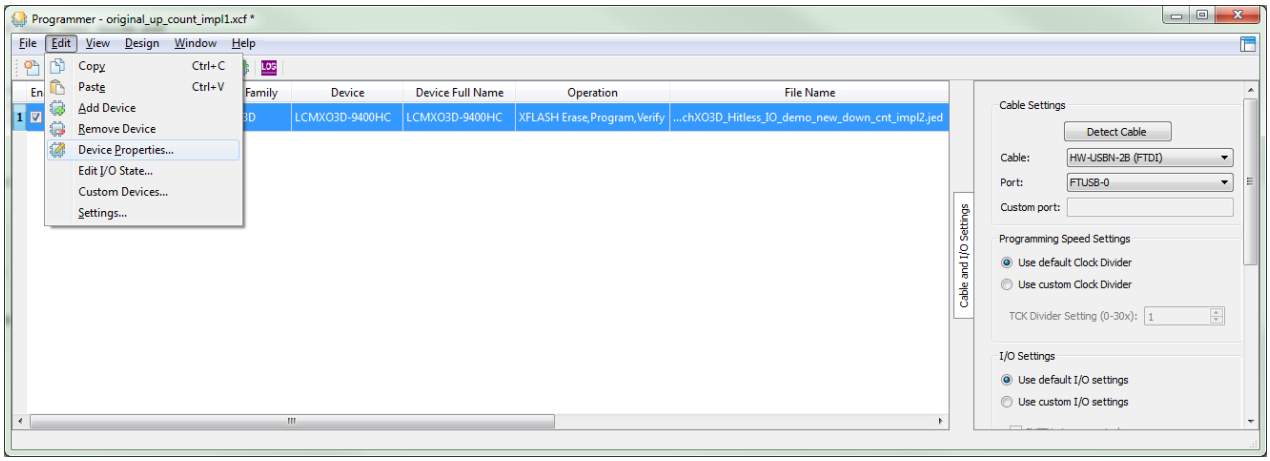

Figure 6.14. Device Properties Option

4. From the **Access mode** drop- down list select **Flash Background Mode** as shown in Figure 6.15.

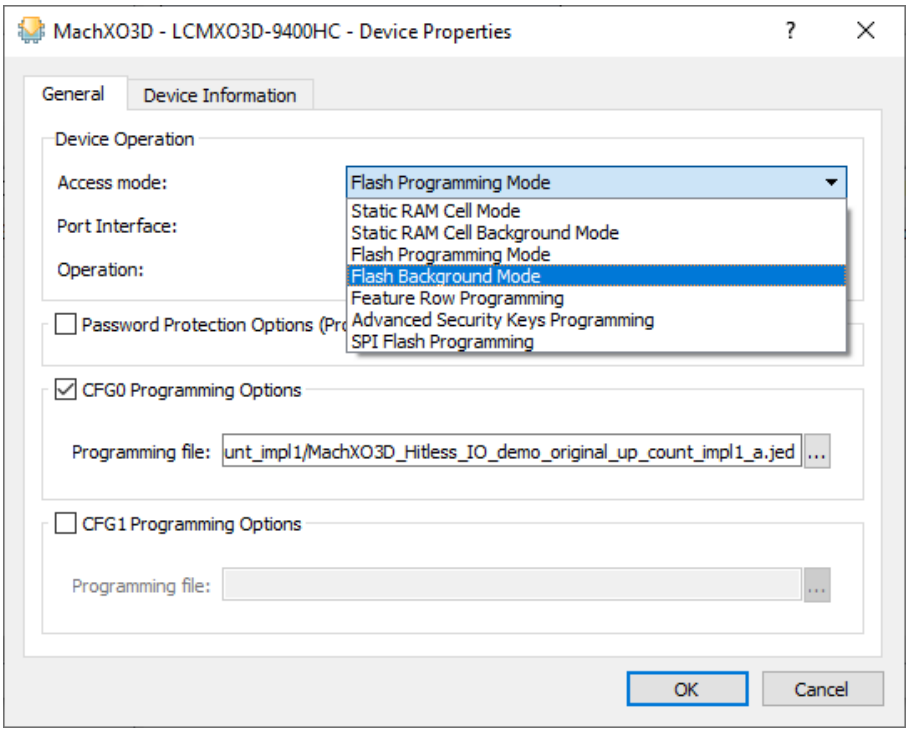

Figure 6.15. Access Mode Options

5. Select **XFLASH TransFR** from the **Operation** dropdown menu (Figure 6.16) to transfer the new bitstream from Configuration Flash to SRAM using the TransFR feature.

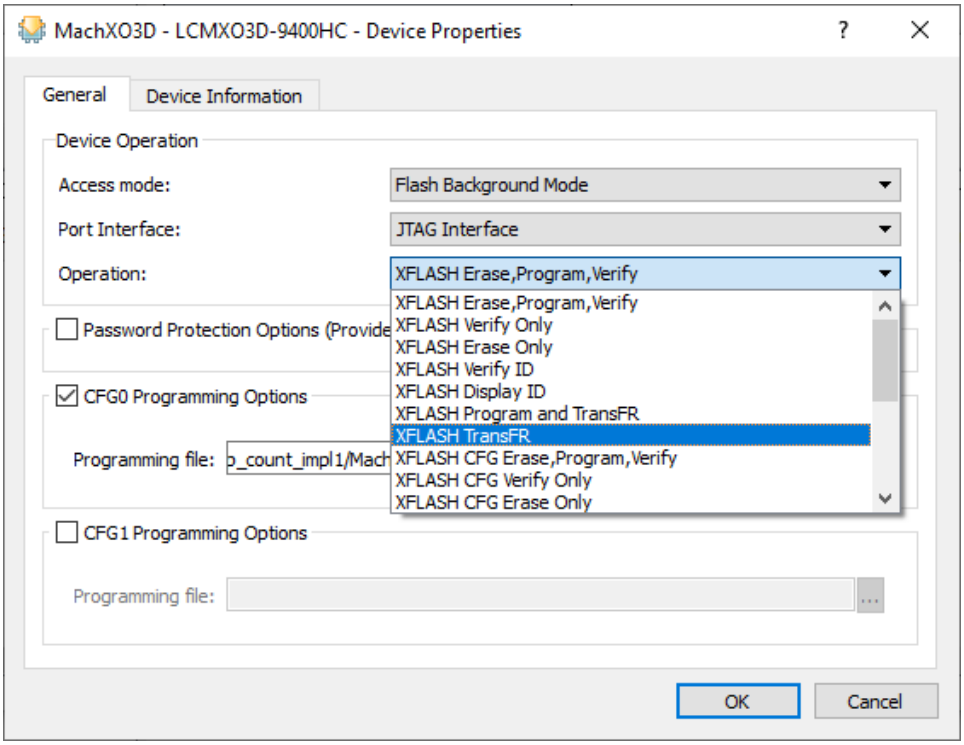

Figure 6.16. Operation Options

6. Select **Design > Program** (Figure 6.17) or click the **Program** button ( ) from the toolbar to execute the Flash-to-SRAM transfer.

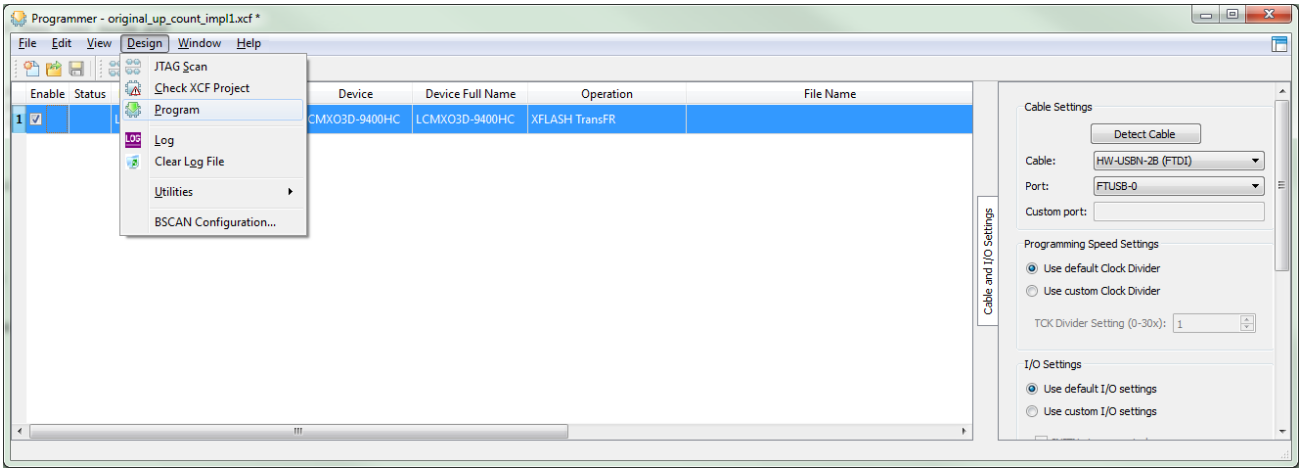

Figure 6.17. Programming the Device

6.5. Resuming Normal Operation

When the Flash-to-SRAM transfer is completed, the new bitstream begins operation seamlessly. LED count starts counting down from the LED count value that is stopped at the Preparing for Update section. D20 keeps increasing by 1 every second. D20 is only driven by EBR context. LED OFF represents 1. LED ON represents 0. D20 increases from 4'h0 to 4'hF. The I/O and EBR context keep steady during background reprogramming and TransFR mode reconfiguration. Not only has the design been successfully updated without changing or glitching the LED output count value and EBR context, the new circuit is able to resume counting from the previous operation point by referring to the held output states.

If reconfiguration mode is not TransFR, or reprogramming mode is not background mode, LED count and D20 are in uncertain state during reprogramming and reconfiguration process.

7. Rebuilding the Design

To rebuild the design:

1. In Lattice Diamond, open the Lattice Design file (*.ldf file) by clicking **Open** in the **Project** tab (Figure 7.1).

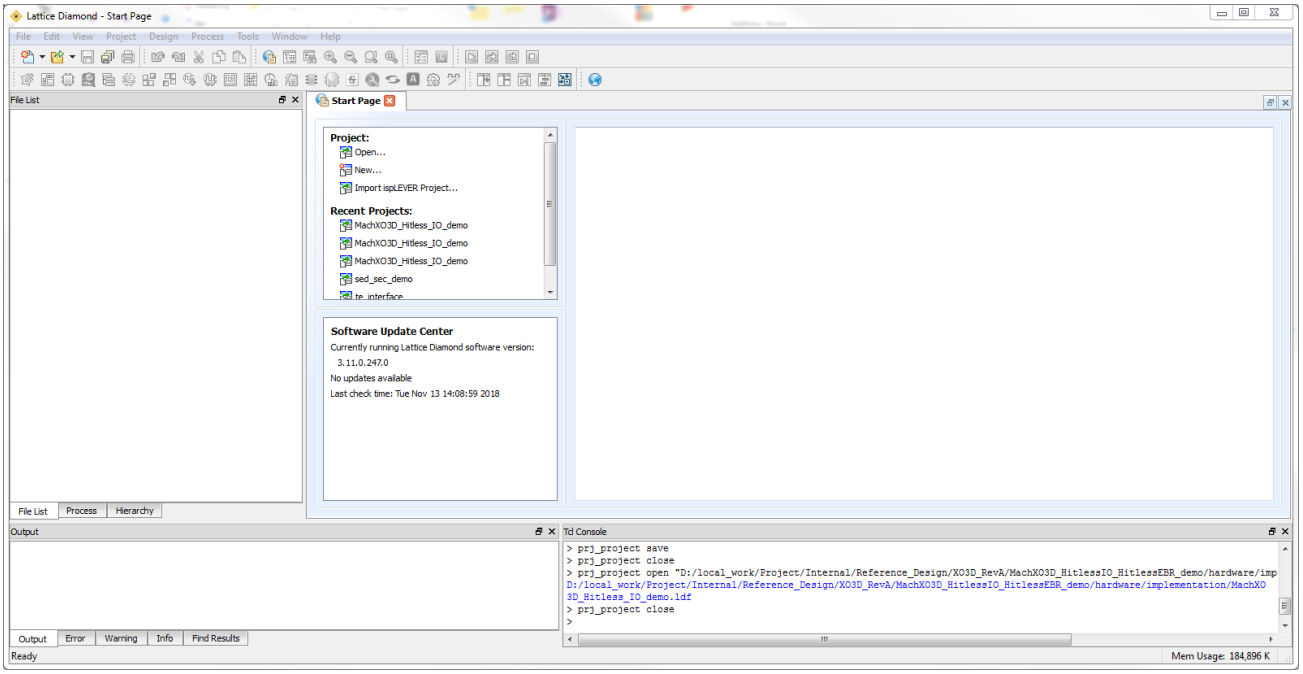

Figure 7.1. Opening Design File

2. There are two implementations in this project: **original_up_count_impl1** and **new_down_cnt_impl2**. Make sure to set the **original_up_count_imp1** implementation active (Figure 7.2).

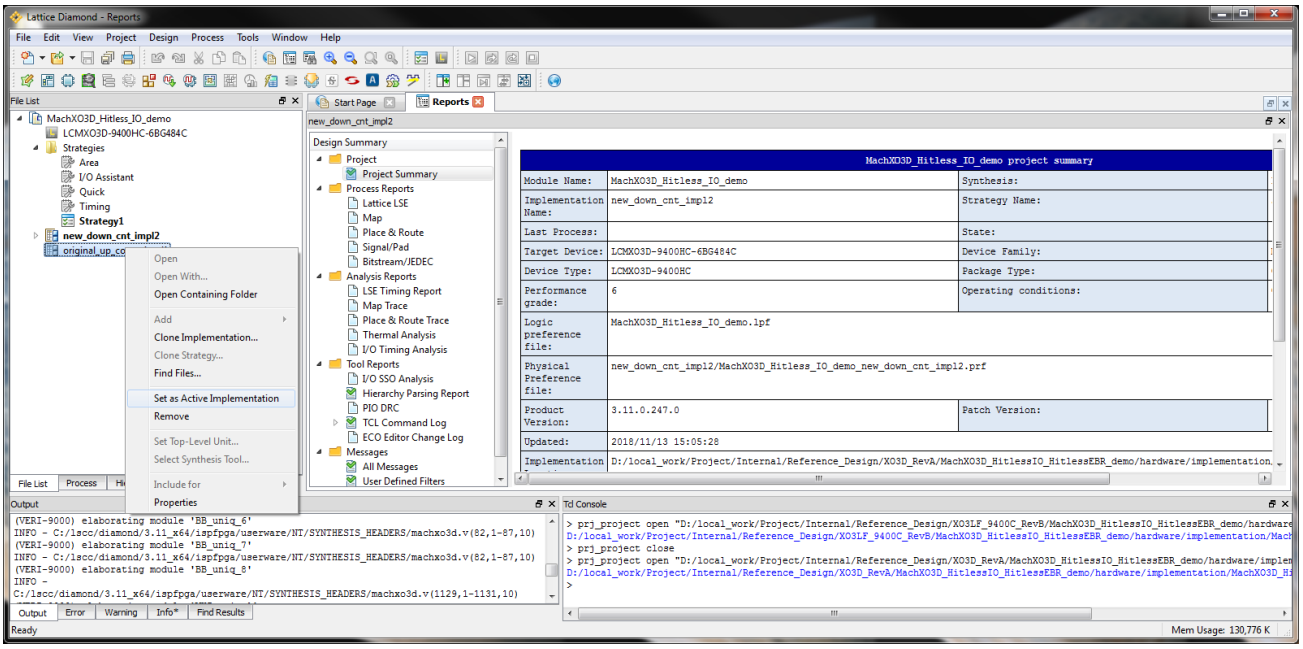

Figure 7.2. Setting original_up_count_impl1 Active

3. Run the **Synthesize Design** process for the design and make sure there are no errors. When you see the green checkbox beside **Synthesize Design** (Figure 7.3), invoke **Spreadsheet View** (Figure 7.4) by clicking the **Spreadsheet View** icon( ) from the toolbar, or select **Tools > Spreadsheet View** from the menu.

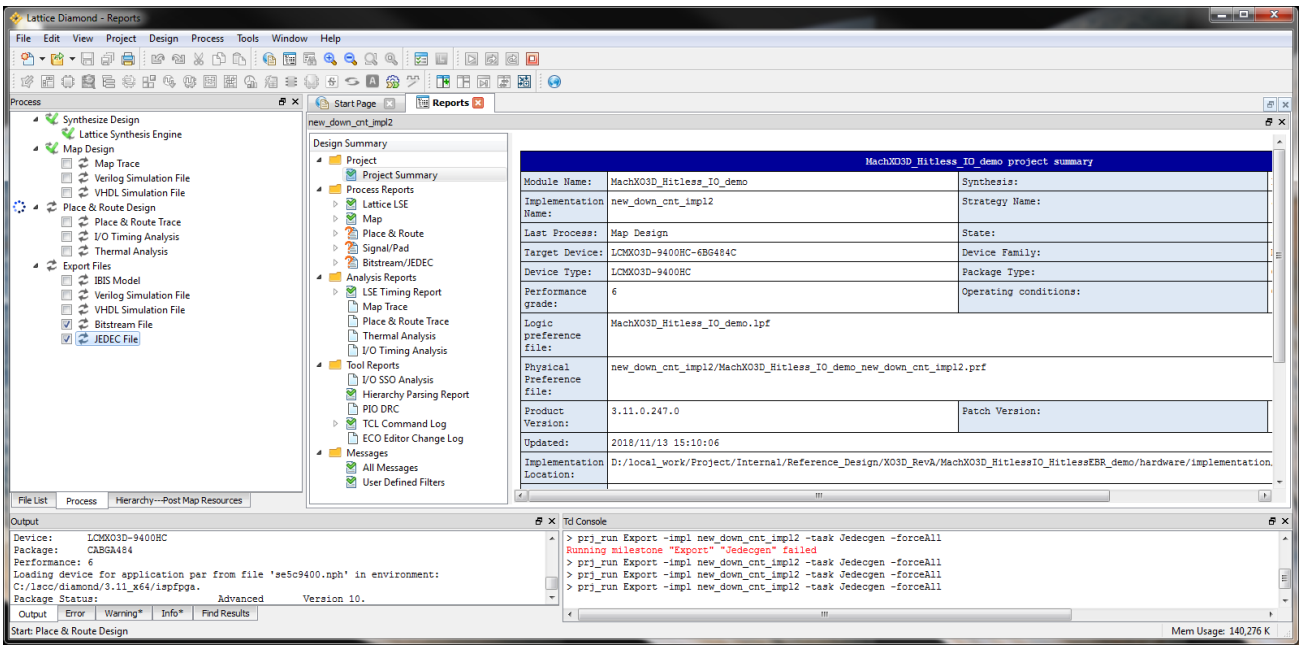

Figure 7.3 Synthesize Design Process

4. In the **Global Preferences** tab of Spreadsheet View, make sure that the **ENABLE_TRANSFR** feature is enabled (Figure 7.4). Save any changes to Spreadsheet View using **File > Save**.

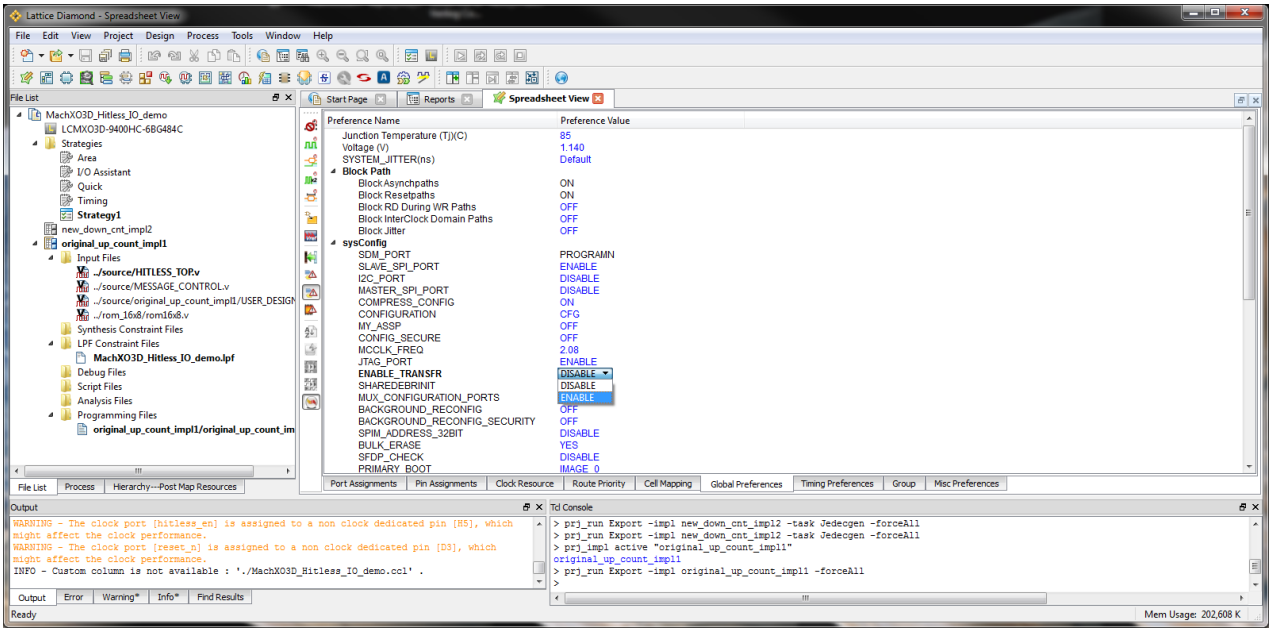

Figure 7.4. Enable Transfer

5. After these preferences are set up, check the checkbox before **Bitstream File** and **JEDEC File** in Process View (Figure 7.5).
6. Double-click **Export Files**, or right-click **Export Files** and choose **Rerun All** to re-run all the previous processes to generate Bitstream file and JEDEC files**.**

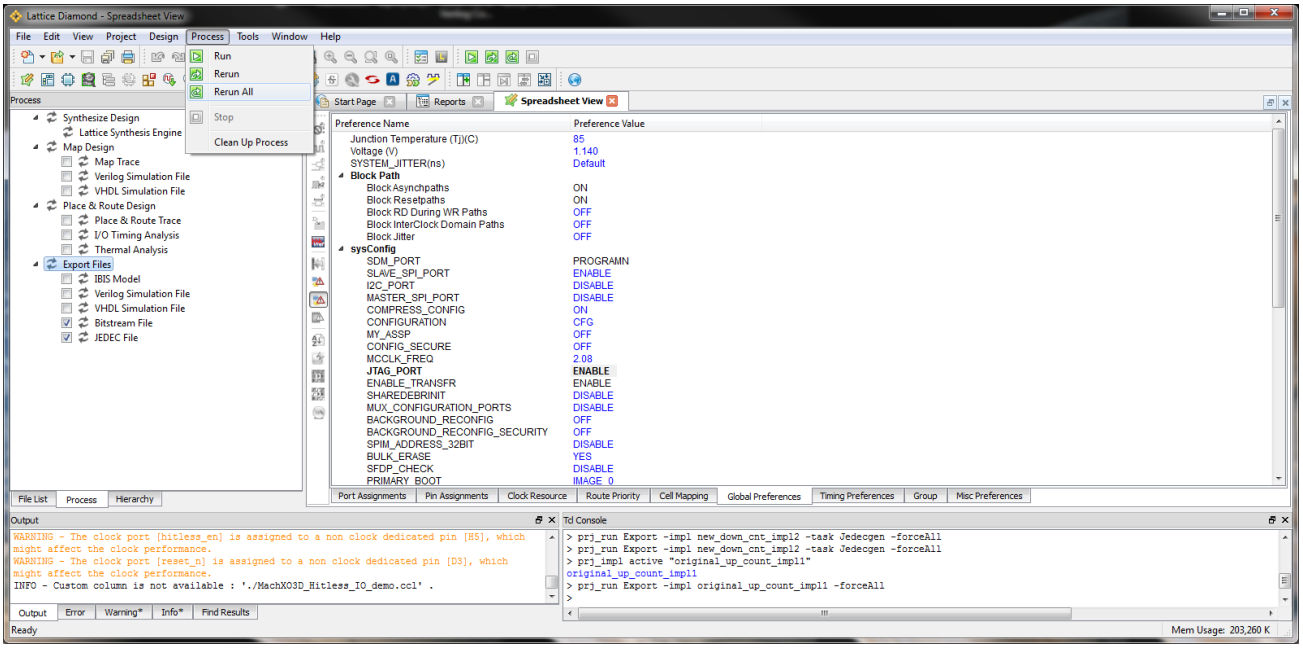

Figure 7.5. Generating the Bitstream

The design rebuilt can be programmed in the MachXO3D development board, as described in the Running the Demo section.

References

Lattice Semiconductor Documents

This is a list of related documents that are available from your Lattice Semiconductor sales representative.

Document	Title
FPGA-DS-02026	MachXO3D Family Data Sheet
FPGA-EB-02020	MachXO3D Development Board User Guide
FPGA-TN-02069	MachXO3D Programming and Configuration Usage Guide

Technical Support

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

Revision History

Revision 1.1, November 2019

Section	Change Summary
Disclaimer	Added this section.
Programming the Device	Updated Figure 6.4.
Updating the Flash Image	Updated Figure 6.7, Figure 6.8, and Figure 6.9.
Updating the Design	Updated Figure 6.15 and Figure 6.16.

Revision 1.0, November 2018

Section	Change Summary
All	Initial release.

www.latticesemi.com