



Hitless Update Demo

User Guide

UG118 Version 1.0

June 2016

Contents

1. Introduction	4
2. Hardware Requirements	4
3. Software Requirements	4
4. Demo Design Overview	4
4.1. Theory of Operation	4
4.2. Hitless Update Demo Design Overview	6
4.2.1. User Design Block	7
4.2.2. Multiplexer Latch to hold Outputs	7
4.2.3. Message Control Block	8
5. Port Assignments and Descriptions	9
6. Demo Package Directory Structure	9
7. Running the Demo	10
7.1. Normal Operation	10
7.2. Update the Flash Image	13
7.3. Prepare for Update	15
7.4. Update the Design	16
7.5. Resume Normal Operation	18
8. Rebuilding the Designs	19
References	22
Technical Support	22
Revision History	23

Figures

Figure 4.1. Hitless Update Demo Design Block Diagram	5
Figure 4.2. Hitless Update Demo Timing Diagram	6
Figure 4.3. TransFR I/O Cell Simplified Block Diagram	7
Figure 4.4. Hitless Update New Top Level Block Diagram	8
Figure 7.1. Diamond Programmer Getting Started Page	10
Figure 7.2. Diamond Programmer Main Interface	11
Figure 7.3. Device Properties Option	11
Figure 7.4. Browse Programming File	12
Figure 7.5. Programming the Device	12
Figure 7.6. Device Properties Option	13
Figure 7.7. Access Mode Options	13
Figure 7.8. Operation Options	14
Figure 7.9. Browse Programming File	14
Figure 7.10. Programming the Device	15
Figure 7.11. MachXO3LF Starter Kit	15
Figure 7.12. Edit I/O State Option	16
Figure 7.13. Edit I/O State Dialog Box	16
Figure 7.14. Device Properties Option	17
Figure 7.15. Access Mode Options	17
Figure 7.16. Operation Options	18
Figure 7.17. Programming the Device	18
Figure 8.1. Opening Design File	19
Figure 8.2. Design	20
Figure 8.3. Port Assignments	21
Figure 8.4. Generating the Bitstream	21

Tables

Table 4.1. Truth Table for Normal_Operation, Hitless_Enable	6
Table 5.1. FPGA Demo Design Ports	9

1. Introduction

In mission critical systems such as data center, storage or networking equipment, feature improvements and bug fixes are performed through background updates. However, up to now, designers have been avoiding updating of the control Programmable Logic Device (which typically performs functions such as power management, reset management, glue logic and other housekeeping functions on the board) as it forced a power cycle or reset of the entire system to enable the new algorithm to take effect.

The Lattice Semiconductor MachXO3™ device, the most popular control PLD, can be updated in the background and the new algorithm takes effect without interrupting board-level operations. There is no need for a power-cycle or reset of the device or the system. This feature is known as the *hitless* or *zero-downtime* system update.

This demo user guide describes the technology and provides the design details of the MachXO3 Hitless Update feature.

To demonstrate the Hitless Update I/O design, the Lattice Diamond® software is used to enable key features such as TransFR and Leave Alone.

2. Hardware Requirements

To demonstrate the Hitless Update I/O design, the following hardware are required:

- PC running Windows 7 Operating System
- MachXO3LF starter kit
- USB Type-A to mini-USB Type-B cable for programming the bitstream

3. Software Requirements

The following software programs are available at www.latticesemi.com/en/Products/DesignSoftwareAndIP

The software programs are available for download only if you log in at www.latticesemi.com

- Lattice Diamond® Design Software, version 3.7 or later.
- Lattice Programmer® Software for bitstream downloading.

4. Demo Design Overview

4.1. Theory of Operation

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 allow critical system control logic and state-machines to be updated in the background without impacting dependent downstream circuits such as power supplies or alarms. The MachXO3 Hitless Update feature is specifically designed to hold critical signal output states stable while the underlying logic is updated. The held output states do not glitch during the update process. The held output states 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 Update design uses silicon features built into the MachXO3 FPGA in concert with minimal additions to the User Design logic. MachXO3 FPGA supports background reprogramming and TransFR mode reconfiguration. A simple multiplexer latch circuit is added to the critical user outputs and a simple common Message Control block is used for interfacing to the System Update Controller. The System Update Controller communicates with the Message Control block over a suitable channel, for example signal wire or I²C, to coordinate the update event.

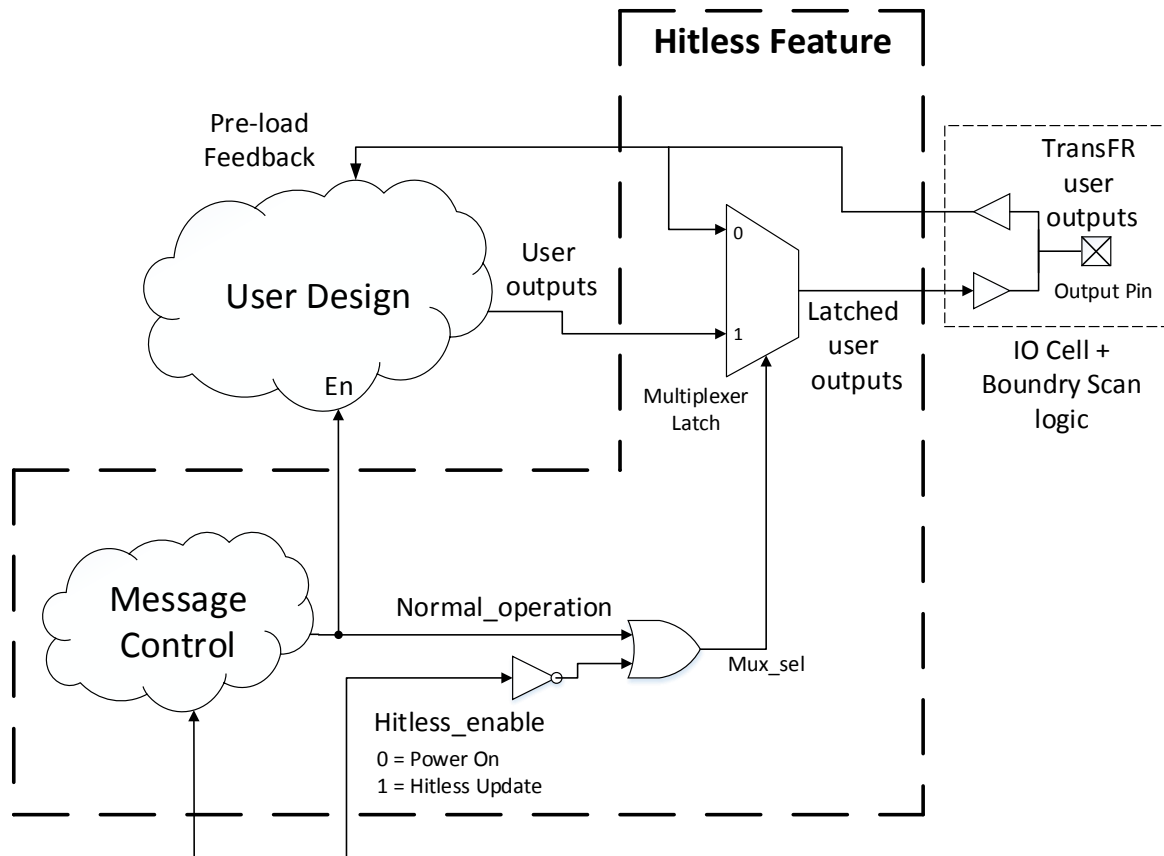


Figure 4.1. Hitless Update Demo Design Block Diagram

The following is a general description of the Hitless Update feature operation. See [Figure 4.1](#) Hitless Update demo design block diagram, and [Figure 4.2](#), Hitless Update demo design timing diagram.

A Normal_operation signal is used to support the Hitless Update 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 Update operation. A message from an external controller is used to de-assert Normal_operation, typically after the new bitstream has been programmed into Flash memory and just prior to a sysConfig REFRESH command or PROGRAMN pin toggle.

The User Design outputs are captured and held with the soft 2:1 multiplexer latch. Normal_operation then remains de-asserted until the MachXO3 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 TN1087 – [Minimizing System Interruption During Configuration Using TransFR Technology](#)). 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.

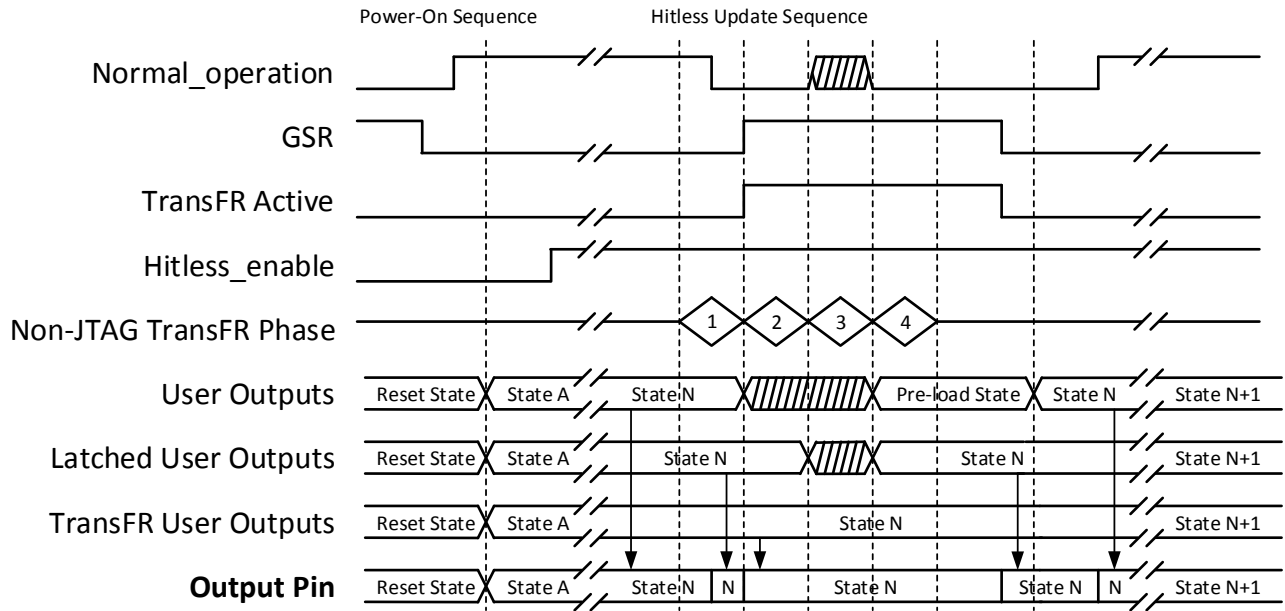


Figure 4.2. Hitless Update Demo Timing Diagram

The Hitless Update 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, then 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 4.1](#) for the truth table showing both the general behavior of the feature and the specific behavior of the demonstration design.

Table 4.1. Truth Table for Normal_Operation, 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 off)
1	0	User Defined Hold/Preload State	Feedback Value

4.2. Hitless Update Demo Design Overview

The onboard LEDs of the MachXO3LF starter kit are used to demonstrate the Hitless Update feature of the MachXO3 device.

As shown in [Figure 4.3](#), the Hitless top level design is divided into three major blocks:

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

For detailed pin/port descriptions, see the [Port Assignments and Descriptions](#) section.

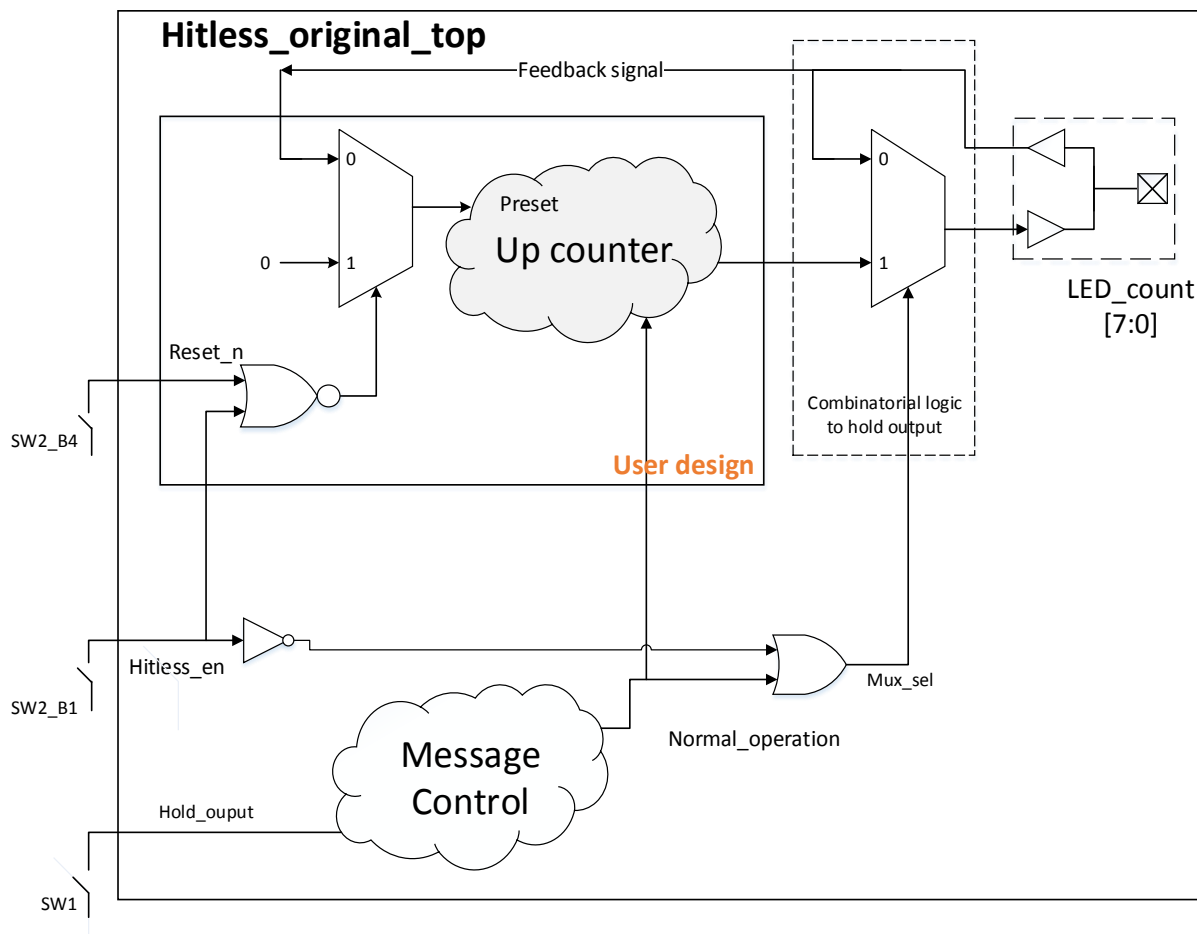


Figure 4.3. TransFR I/O Cell Simplified Block Diagram

4.2.1. User Design Block

This block contains the original user logic, in this case a simple 8-bit up-counter design. The original user logic is updated to include two accommodations which support the hitless update 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.

4.2.2. Multiplexer Latch to hold Outputs

The hitless update design addition consists of 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.

4.2.3. Message Control Block

This hitless update design addition is used to communicate messages to the hitless top level design that the transFR operation is about to occur. SW1 represents the ‘external controller’ as described in the [Theory of Operation](#) section. The message is conveyed by closing SW1, a momentary switch on the XO3LF Break-out board. The closing edge (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.

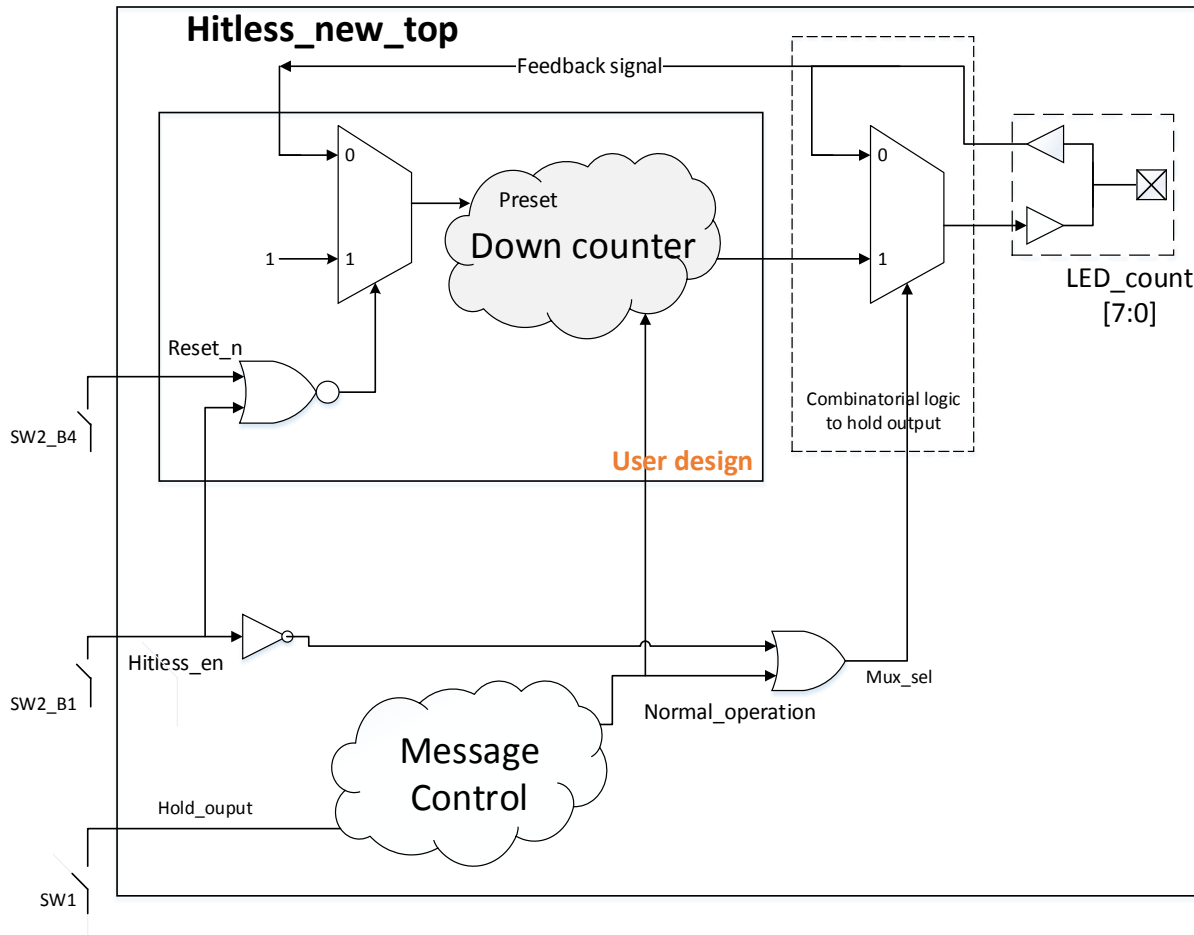


Figure 4.4. Hitless Update New Top Level Block Diagram

A second, nearly identical design project is included as part of the Hitless Update demonstration. As shown in [Figure 4.4](#), only the user design block is changed to include a Down Counter block in place of the prior Up Counter. All the other blocks remain untouched.

The user will be able to switch between the designs without causing glitches or state changes to the LED outputs.

5. Port Assignments and Descriptions

Table 5.1. FPGA Demo Design Ports

Port Name	I/O Type	Width	Description
Reset_n	Input	1	Asynchronous reset, active low (SW2 position 4)
Clk	Input	1	Input clock from crystal
Hitless_en	Input	1	Toggle switch (SW2 position 1) to differentiate between the power on and hitless update operation. Down (0) – Power on Up (1) – Hitless Update
hold_output	Input	1	Push-button switch (SW1) to hold the output state
LED_count	Inout	8	LEDs indicating the output values

6. Demo Package Directory Structure

machXO3LF_hitless_update_demo_design

- bitstream (contains the .jed files for the demonstration)
- docs (contains the user guide UG118 “Hitless_Update_Demo” for the demonstration)
- hardware
 - Source (hdl file for the demonstration)
 - HITLESS_TOP.v
 - MESSAGE_CONTROL.v
 - New_down_count_impl2
 - USER_DESIGN.v
 - Original_up_count_impl1
 - USER_DESIGN.v
 - Implementation
 - New_down_cnt_impl2
 - original_up_count_impl1
 - MachXO3LF_Hitless_Update_demo.ldf
 - MachXO3LF_Hitless_Update_demo.lpf

7. Running the Demo

The Hitless Update demo consists of five parts:

1. Normal Operation – Load the ‘original’ design. The output LEDs are under the ‘original’ design control.
2. Update the Flash image – Program the ‘new’ design into Configuration Flash in the background.
3. Prepare for Update – Communicate to the device regarding a design update using an external switch. The binary counter value is frozen.
4. Update the Design – Reconfigure the configuration SRAM with the new bitstream image contained in the configuration Flash array using the TransFR and Leave Alone features. LED outputs are held stable without glitching.
5. Resume Normal Operation – The updated ‘new’ design is active with the outputs driven by the updated design.

7.1. Normal Operation

Loading the initial design. The onboard output LEDs start counting upwards.

1. Open the Diamond Programmer application. The Getting Started page opens as shown in [Figure 7.1](#).

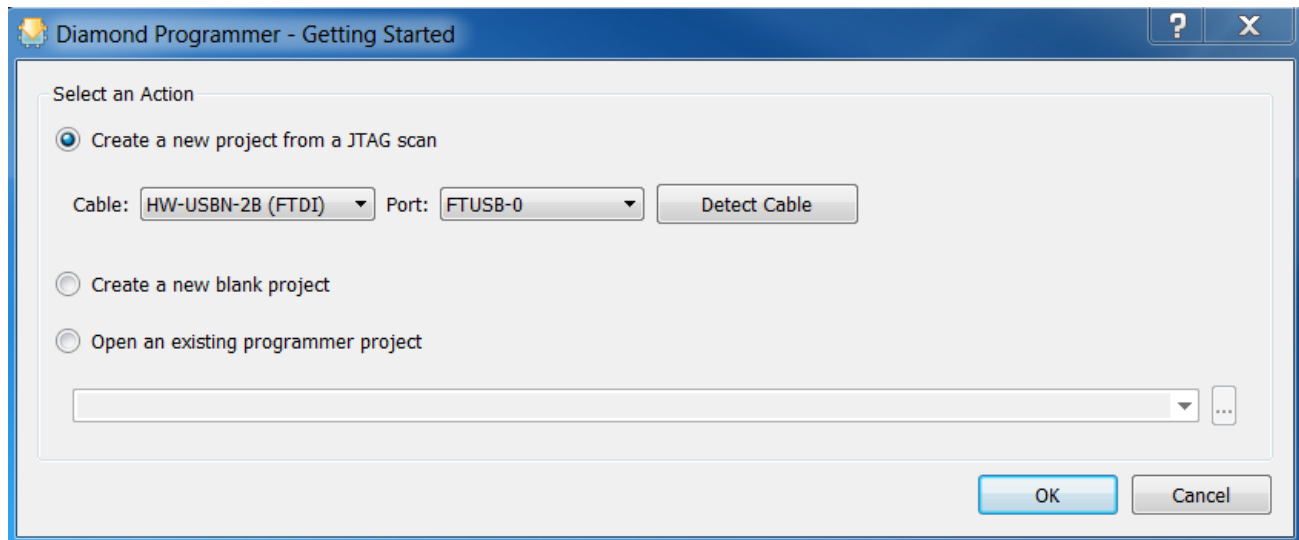


Figure 7.1. Diamond Programmer Getting Started Page

2. Click **OK** and the Diamond Programmer starts scanning the board attached to the USB cable.

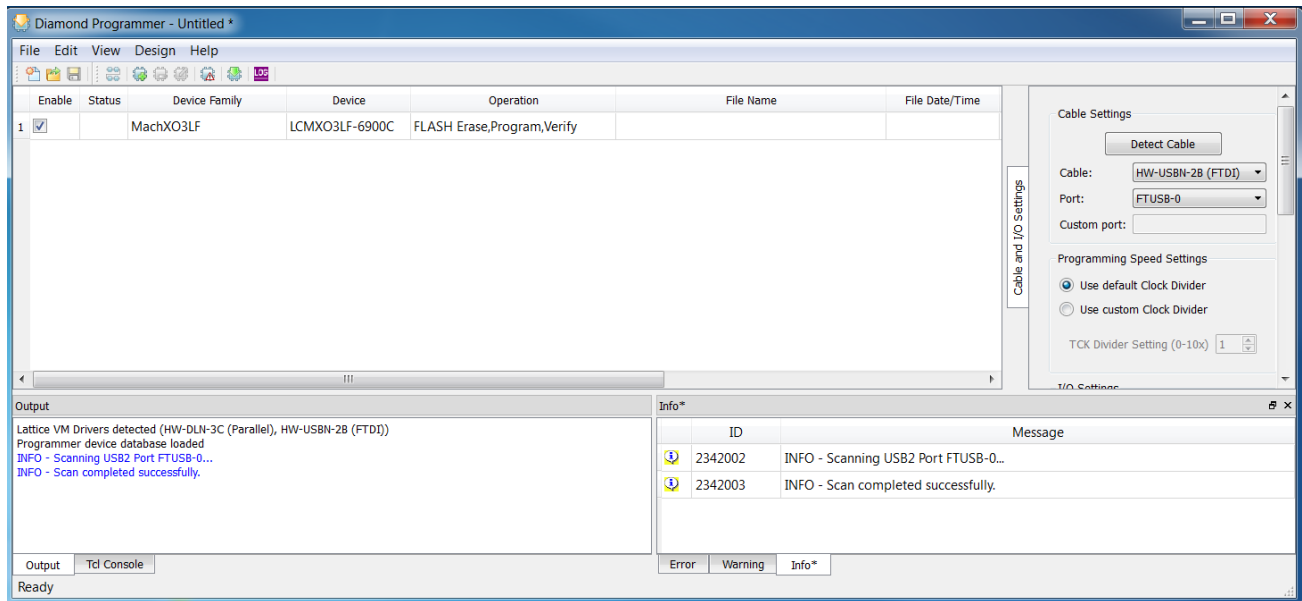


Figure 7.2. Diamond Programmer Main Interface

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

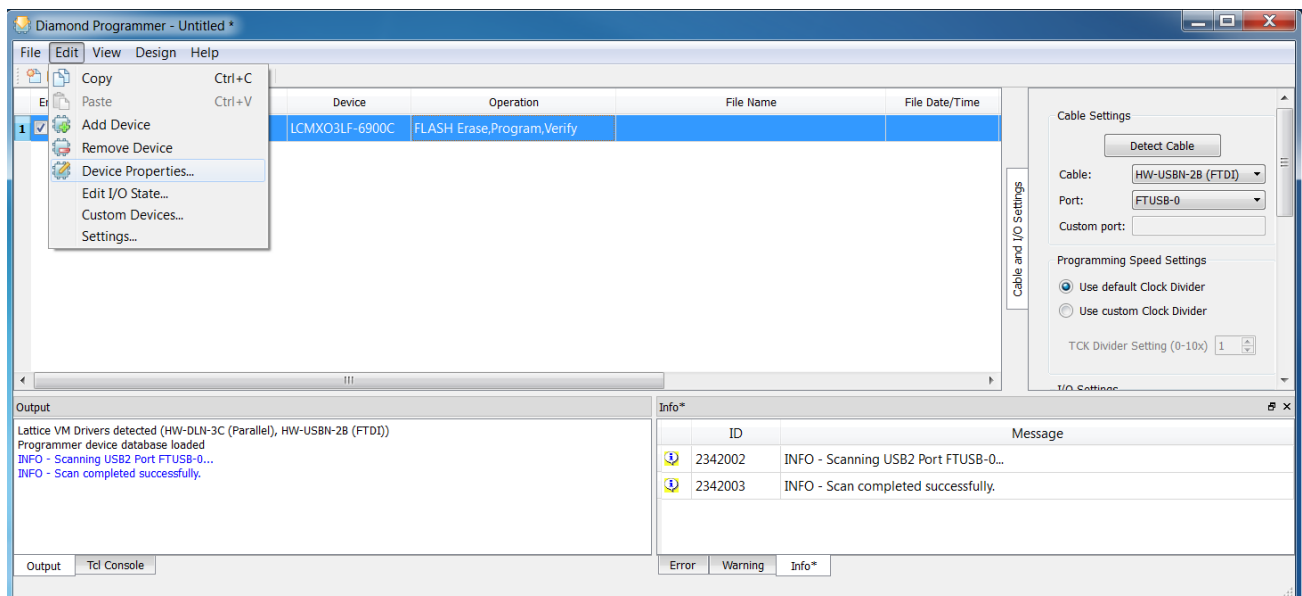


Figure 7.3. Device Properties Option

- Confirm that Access Mode is set to 'Flash Programming Mode' and Operation is set to 'FLASH Erase, Program, Verify.' Under **Programming Options**, navigate to the \bitstream folder and select the 'Original' bitstream (MachXO3LF_Hitless_Update_demo_original_up_count_impl1.jed) to program into the MachXO3LF device by clicking the Browse button.

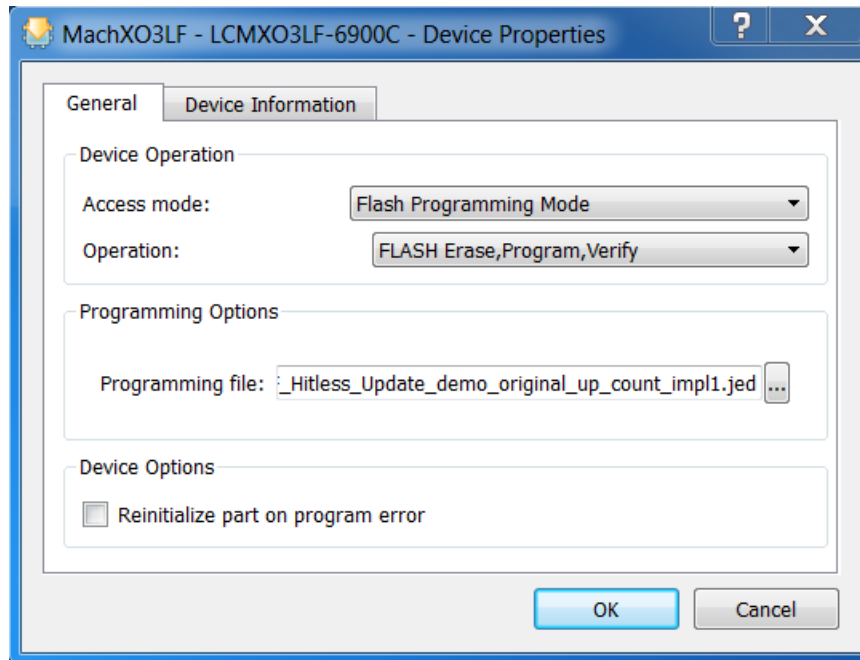


Figure 7.4. Browse Programming File

The next step is to program the device.

5. From the menu bar, select **Design > Program**.

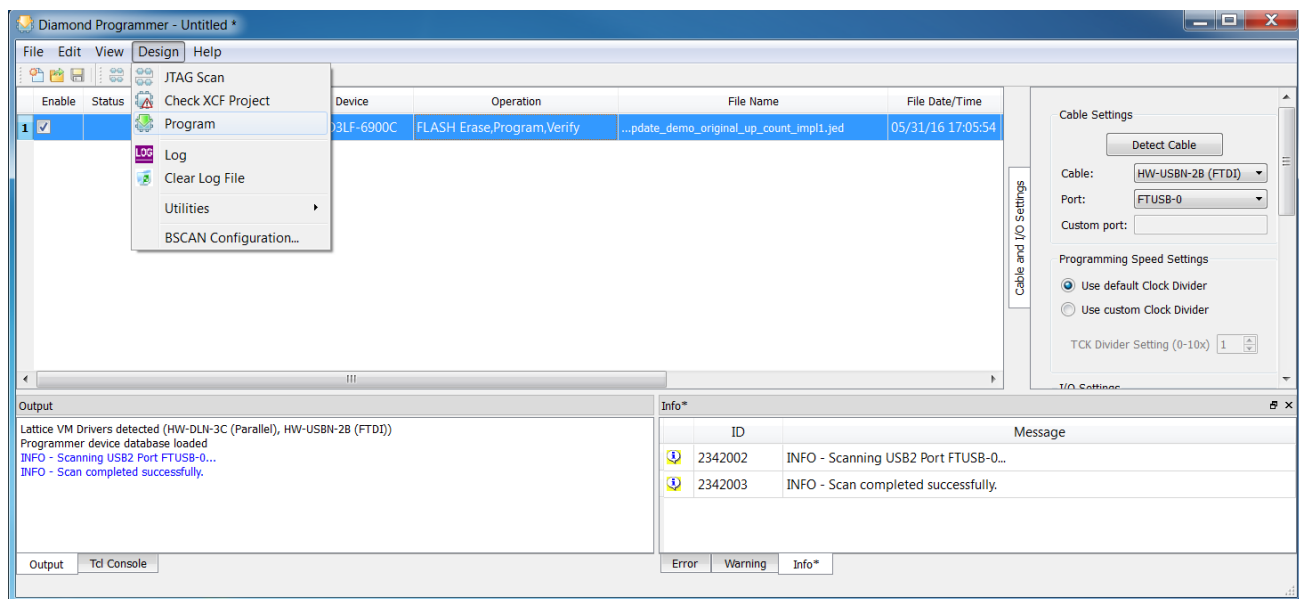


Figure 7.5. Programming the Device

When the programming of the device is completed, the LEDs on the MachXO3LF starter kit start counting up from zero.

7.2. Update the Flash Image

(Background programming. Outputs will not glitch or change state.)

1. 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.

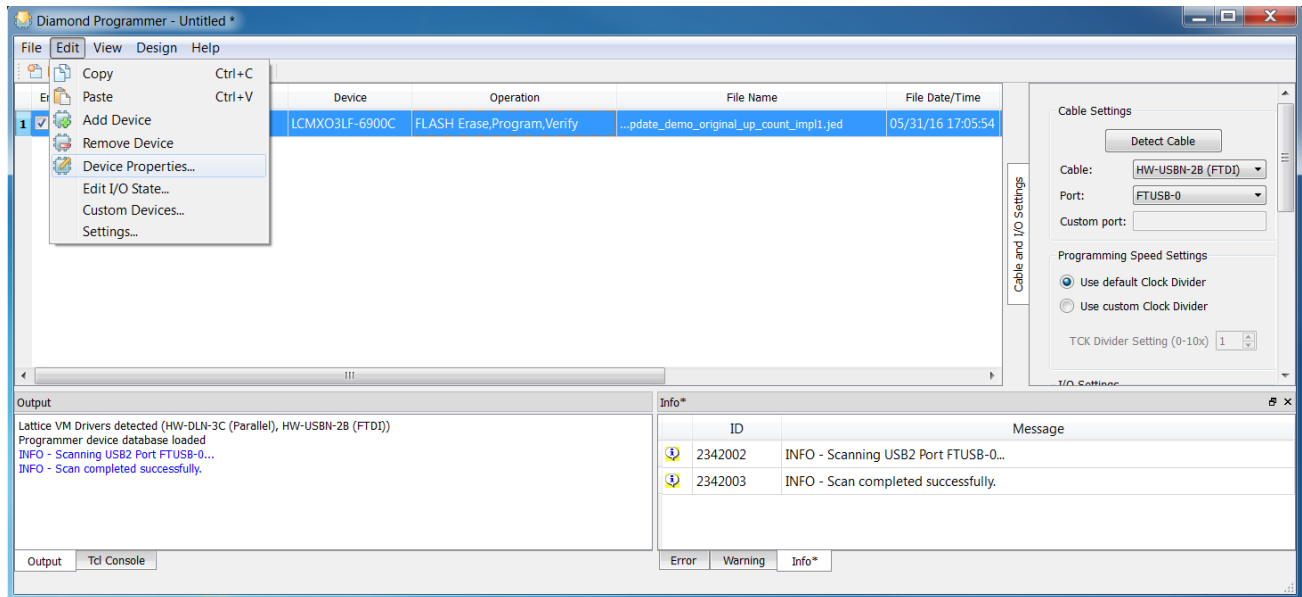


Figure 7.6. Device Properties Option

2. From the **Access Mode** drop-down list select **Flash Background Mode** as shown below.

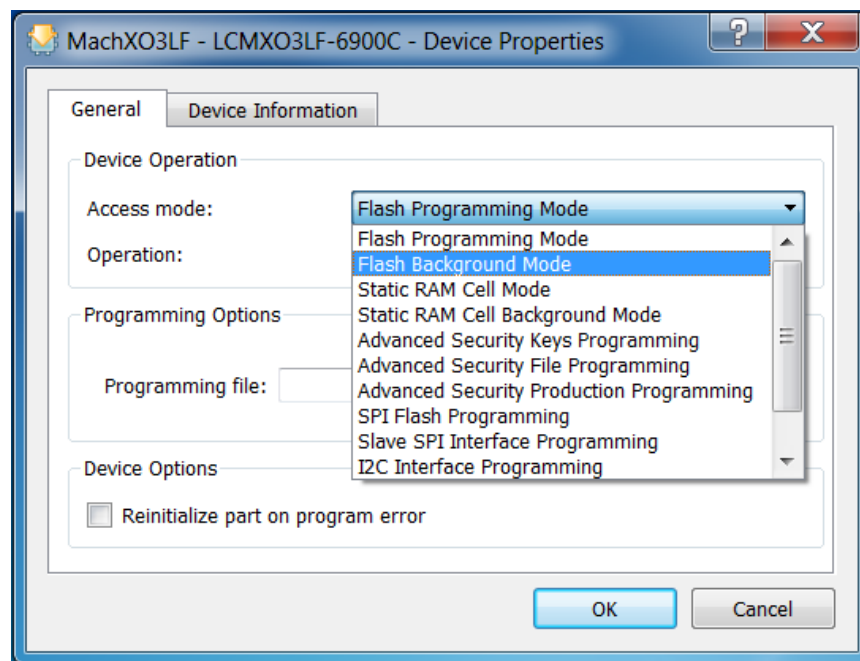


Figure 7.7. Access Mode Options

3. From the **Operation** drop-down list select **XFLASH Erase, Program, Verify**.

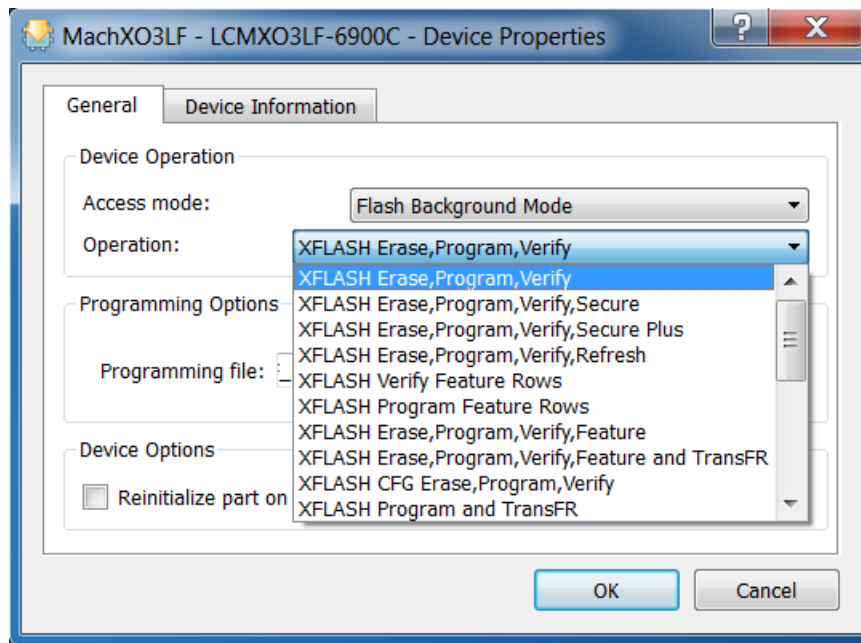


Figure 7.8. Operation Options

4. Under Programming Options, navigate to the \bitstream folder and select the 'new' programming file (MachXO3LF_Hitless_Update_demo_new_down_cnt_impl2.jed) by clicking the Browse button.

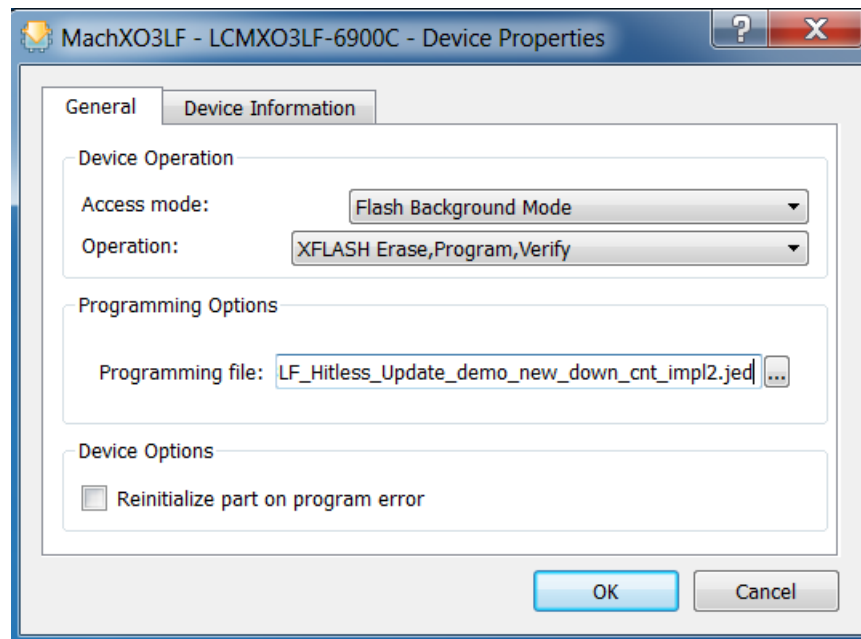


Figure 7.9. Browse Programming File

The next step is to program this bitstream into the device.

5. From the menu bar, select **Design > Program**.

During the Program operation, the LEDs do not change state or glitch. This can be confirmed using an oscilloscope if desired.

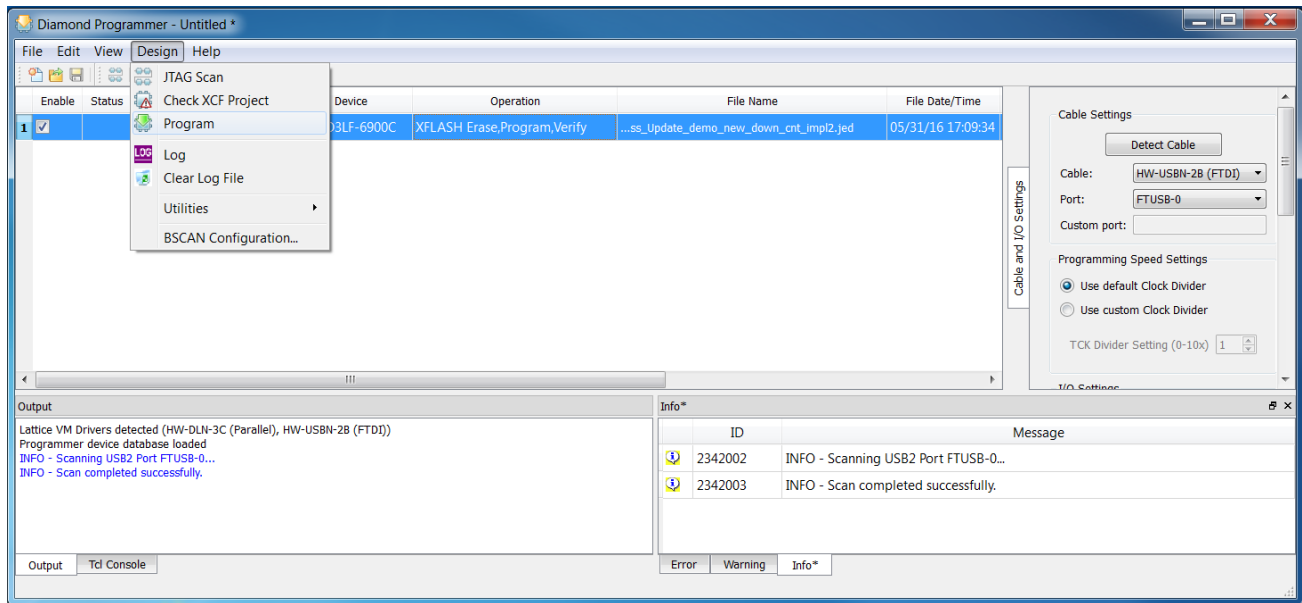


Figure 7.10. Programming the Device

7.3. Prepare 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 positions of SW2 (see Figure 7.11) are 'Up', then press the SW1 switch momentarily to freeze the LED binary count output on D2-D9. Note that if SW2 position 1 is down, the counter resets to the power up state when SW1 is pressed. If SW1 is not pressed to freeze the outputs, the outputs may glitch during the update step.

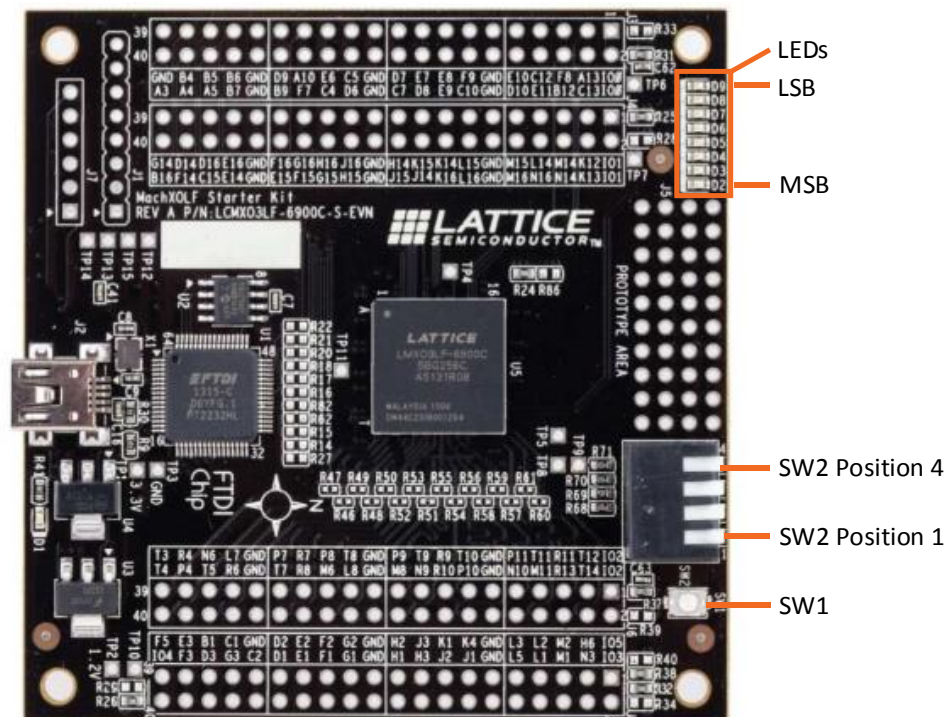


Figure 7.11. MachXO3LF Starter Kit

7.4. Update the Design

(TransFR and Leave Alone. Outputs will not glitch or change state.)

1. Highlight the device row by clicking anywhere on the row. From the menu bar select **Edit > Edit I/O State**.

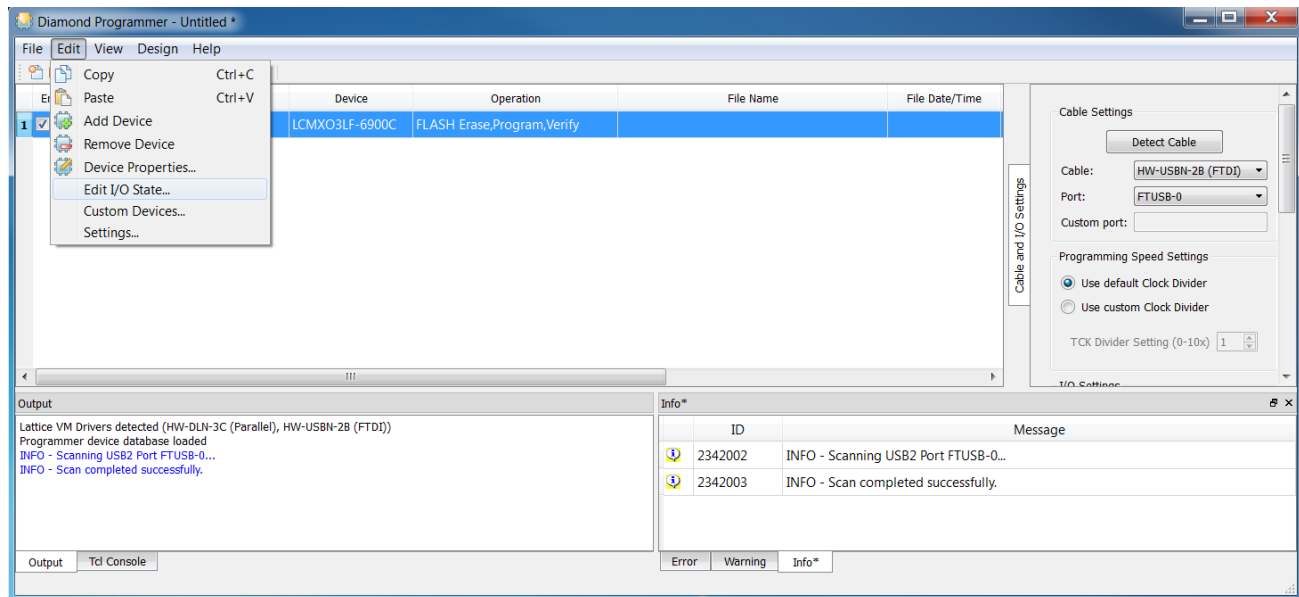


Figure 7.12. Edit I/O State Option

2. In the **Edit I/O State** dialog box, from the I/O state drop-down list select **Leave Alone** as shown in Figure 7.13. Click **OK** to confirm this selection.

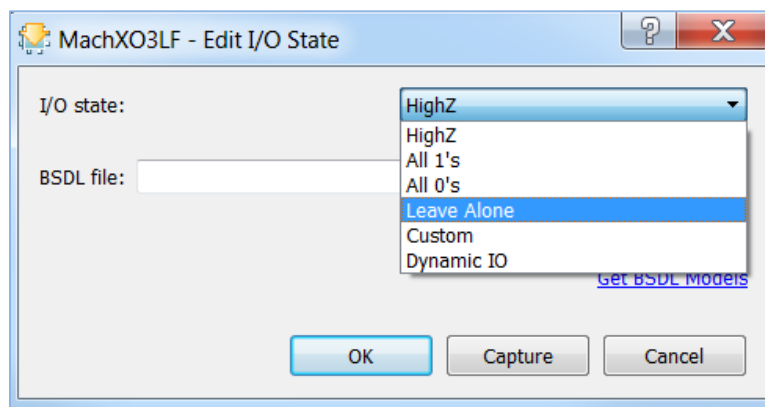


Figure 7.13. Edit I/O State Dialog Box

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.

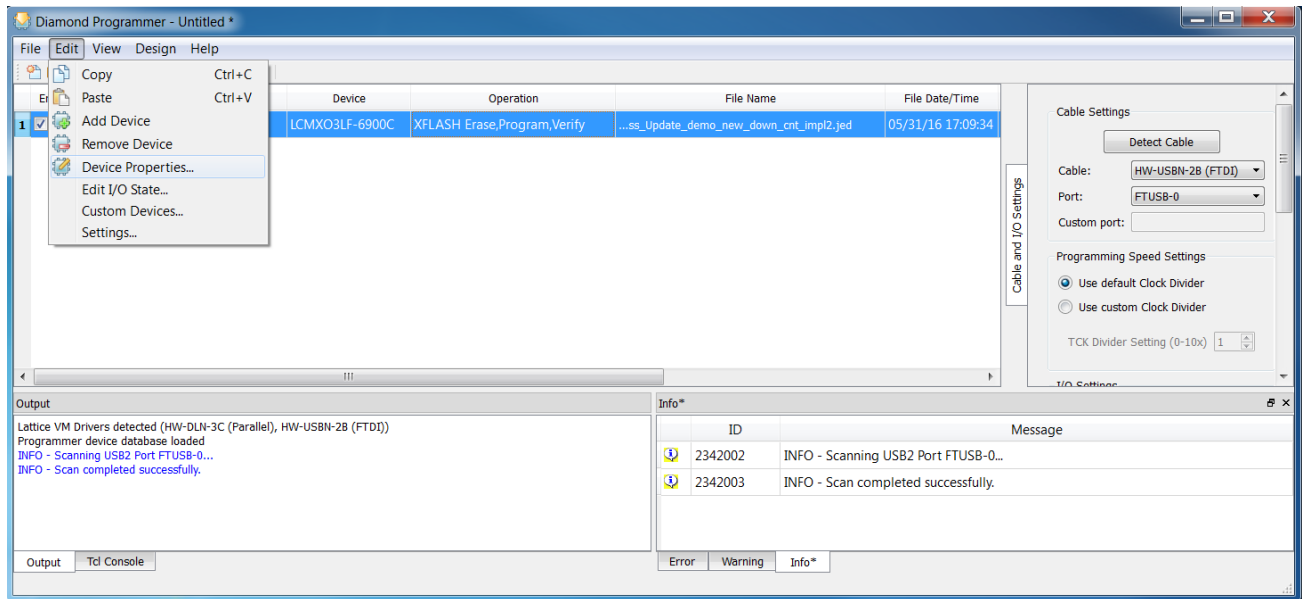


Figure 7.14. Device Properties Option

- From the **Access Mode** drop-down list select **Flash Background Mode** as shown below.

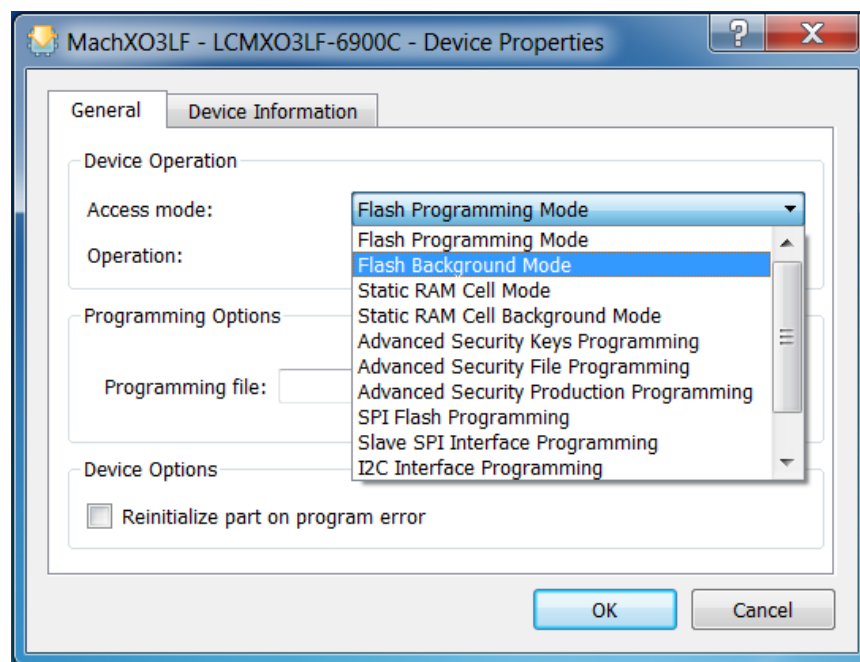


Figure 7.15. Access Mode Options

- Under Operation, select **XFLASH TransFR** to transfer the 'new' bitstream from Configuration Flash to SRAM using the TransFR feature.

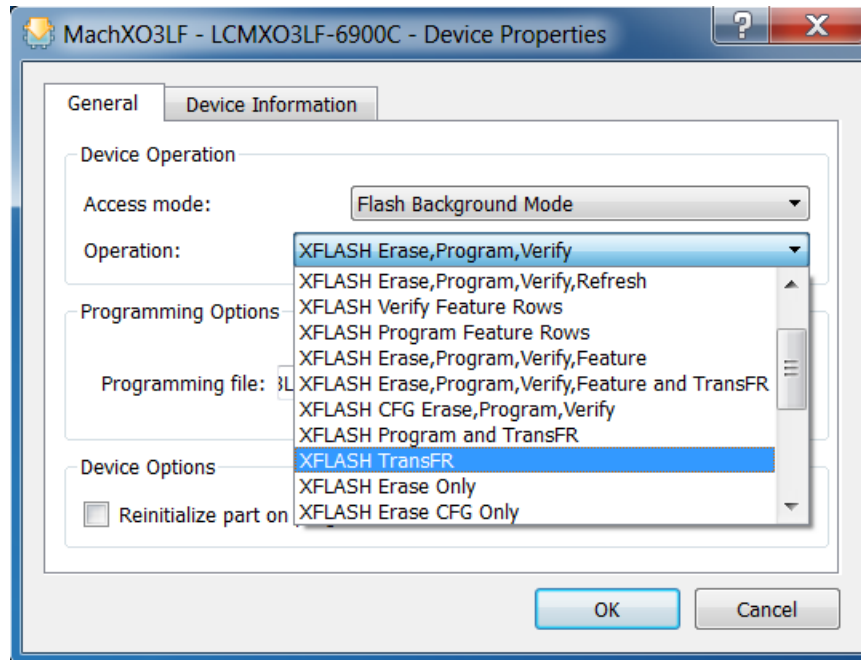


Figure 7.16. Operation Options

- From the menu bar, select **Design > Program** to execute the Flash-to-SRAM transfer. During the TransFR operation, the LEDs do not change state or glitch. This can be confirmed using an oscilloscope if desired.

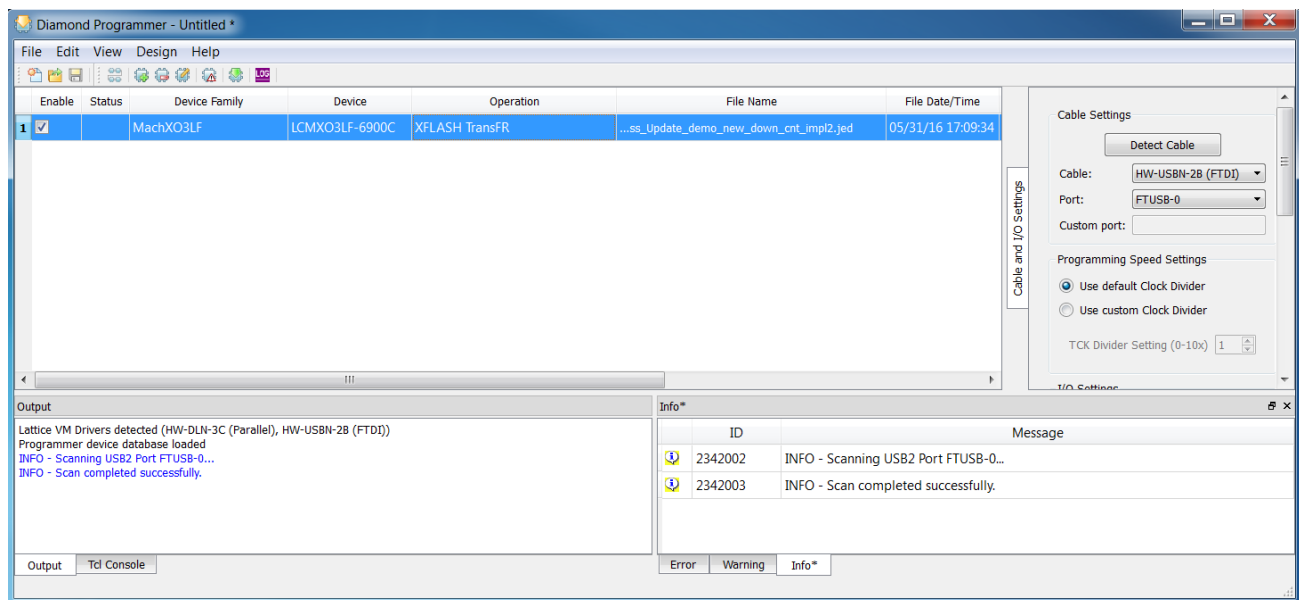


Figure 7.17. Programming the Device

7.5. Resume Normal Operation

When TransFR is complete, the new bitstream begins operation seamlessly, starting from the LED count value preserved in the [Prepare for Update](#) section. Not only has the design been successfully updated without changing or glitching the LED output count value, the new circuit is able to resume counting from the previous operation point by referring to the held output states.

8. Rebuilding the Designs

1. Open the Lattice Design file (*.Idf file) by clicking **Open** under Project.

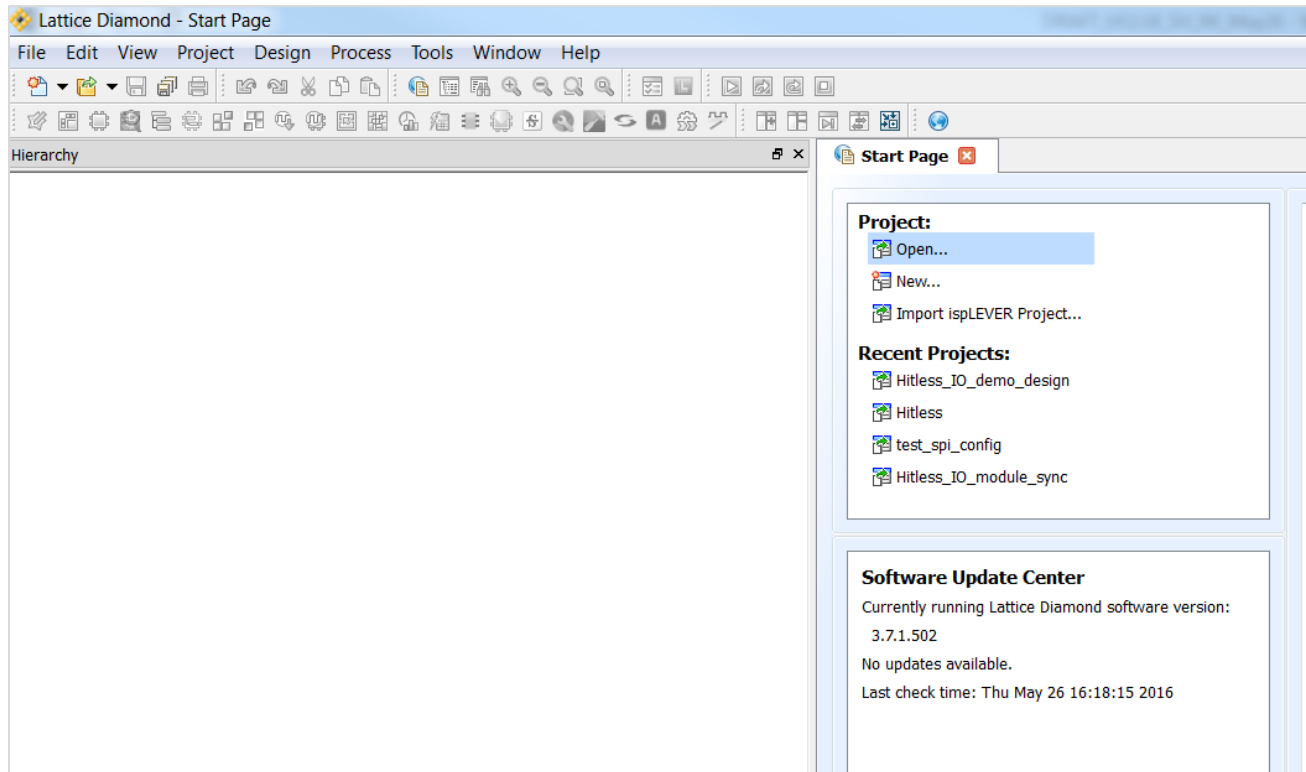


Figure 8.1. Opening Design File

2. There are two implementations (original_up_count_impl1 and new_down_cnt_impl2) in this project. Make sure to set the appropriate implementation to active.

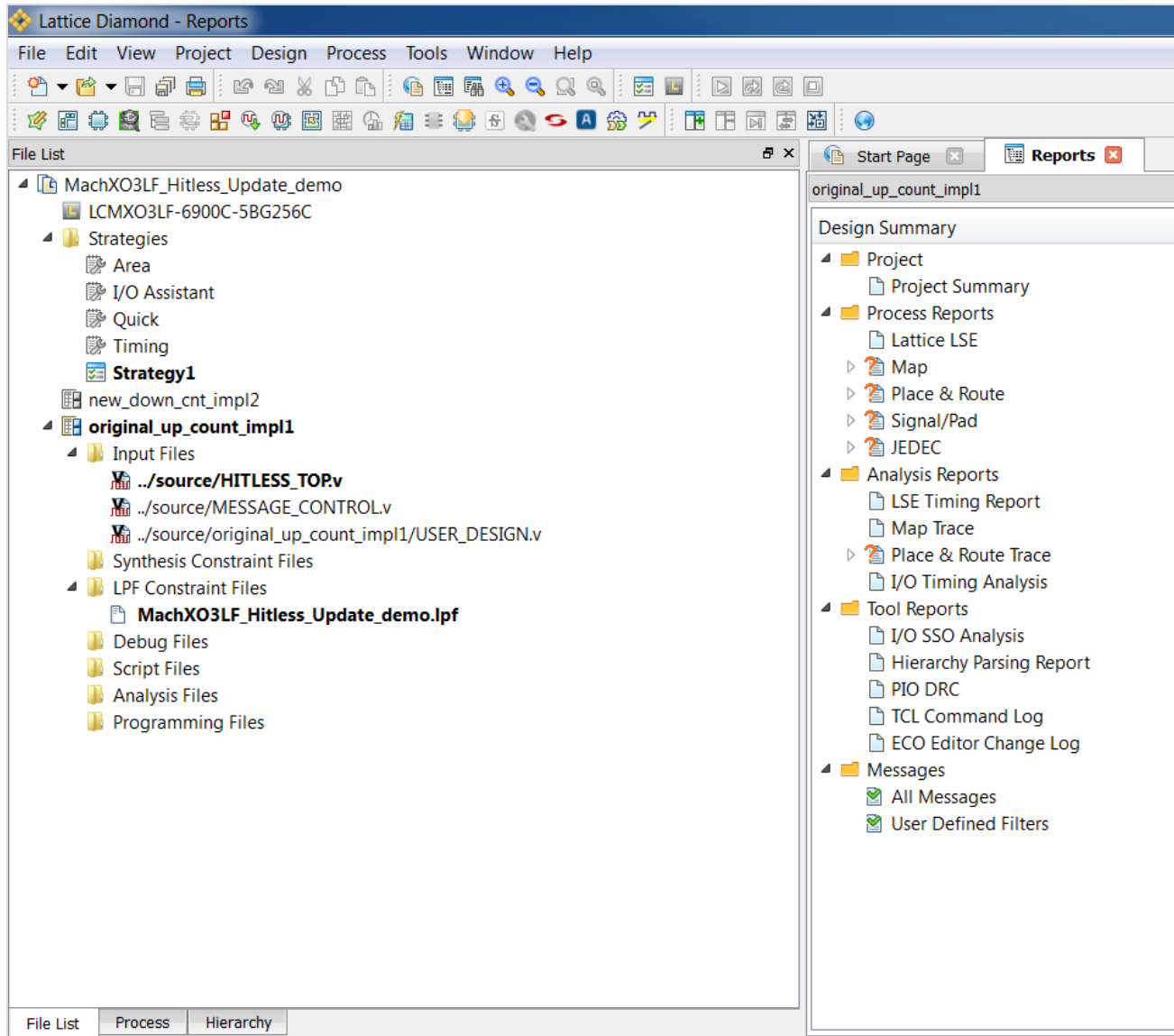


Figure 8.2. Design

- Run the Synthesize phase on the design to make sure there are no errors in the design. When you see the green checkbox besides **Synthesize Design**, click **Spreadsheet View**, go to **Global Preferences** tab and make sure that the **ENABLE_TRANSFR** feature is enabled. Save any changes to Spreadsheet View using **File > Save**.

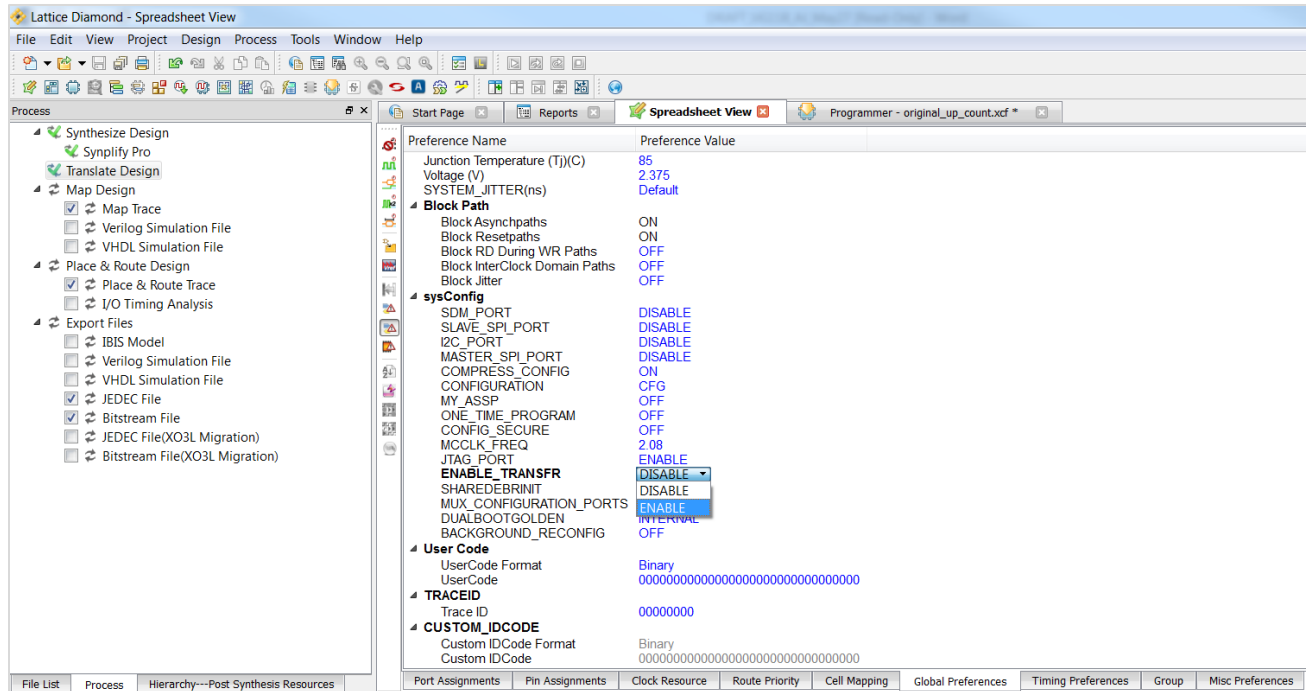


Figure 8.3. Port Assignments

- After these preferences are set up, run the project to generate the bitstream. From the menu bar, select **Process > Run** to run the entire process.

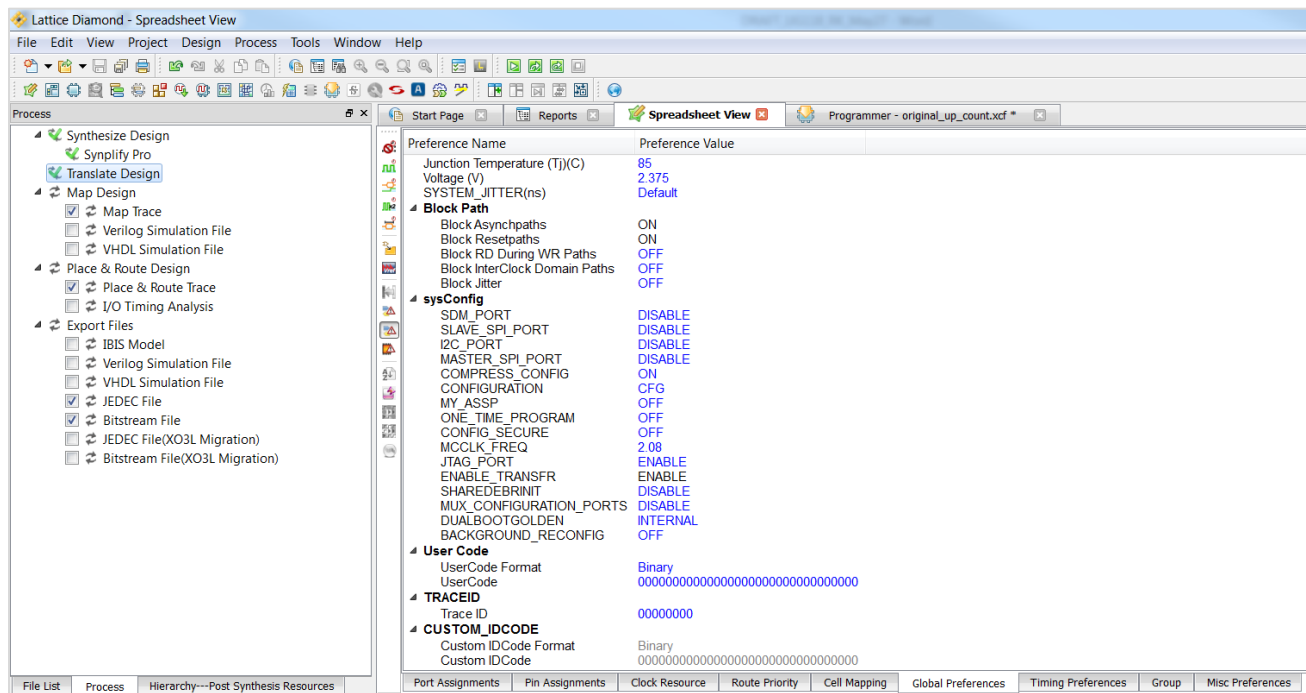


Figure 8.4. Generating the Bitstream

The rebuilt design can be programmed in the FPGA as described in the [Running the Demo](#) section.

References

DS1047 – [MachXO3 Family Data Sheet](#)

EB95 – [MachXO3 Starter Kit User's Guide](#)

TN1087 – [Minimizing System Interruption during Configuration using TransFR Technology](#)

TN1279 – [MachXO3 Programming and Configuration Usage Guide](#)

Technical Support

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

Revision History

Revision 1.0, June 2016

First release.



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