MachXO5-NX I2C Reference Design User Guide

Reference Design

FPGA-RD-02299-1.0

January 2025

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviations	Definition
AHBL	Advanced High-Performance Bus Lite
APB	Advanced Peripheral Bus
BSP	Board Support Package
CPU	Central Processing Unit
FIFO	First In, First Out
GPIO	General Purpose Input/Output
GUI	Graphical User Interface
I/O	Input/Output
I2C	Inter-Integrated Circuit
IFC	Internal Flash Controller
IP	Intellectual Property
LED	Light-Emitting Diode
LMMI	Lattice Memory Mapped Interface
MC	Micro-Controller
PLL	Phase-Locked Loop
QSIM	QuestaSim simulator
RISC-V	Reduced Instruction Set Computer Five
Rx	Receiver
SDK	Software Development Kit
SoC	System-on-Chip
Tx	Transmitter
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus

1. Introduction

This document is the user guide for the Lattice MachXO5-NX™ I2C reference design, which provides the data access to the MachXO5-NX internal flash through the user I2C bus interface.

1.1. Quick Facts

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

Table 1.1. Summary of the Reference Design

General	Target Device Family	MachXO5-NX
	Source Code Format	Verilog, and C Lattice Radiant™ and Lattice Propel™ projects
Simulation	Functional Simulation	Performed
	Timing Simulation	Not performed
	Test Bench	Available
	Test Bench Format	QSIM testbench
Software Requirements	Software Tool and Version	Lattice Radiant 2024.1 Lattice Propel 2024.1
	IP Version (if applicable)	RISC-V MC version 2.6.0 System Memory version 2.2.0 GPIO version 1.6.2 Internal Flash Controller version 2.0.0 I2C Target version 2.1.1 UART version 1.3.0 PLL version 1.9.0 AHBL Interconnect version 1.3.1 AHBL-to-APB Bridge version 1.1.1 APB Interconnect version 1.2.1
Hardware Requirements	Board	Lattice MachXO5-NX-25 development board
	Cable	USB cable for programming

1.2. Features

Key features of the MachXO5-NX I2C reference design include:

- User I2C target for host access:
  - I2C commands to program the MachXO5-NX flash.
- Internal flash access for the MachXO5-NX flash:
  - Erase, write, and read on the CFG, UFM, and USERDATA partitions.

1.3. Naming Conventions

1.3.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL.

1.3.2. Signal Names

- *_n* are active low (asserted when value is logic 0)
- *_i* are input signals
- *_o* are output signals

2. Directory Structure and Files

The directory structure of the MachXO5-NX I2C reference design is shown in the figure below:

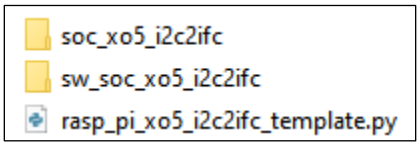

Figure 2.1. Directory Structure

The files included in the MachXO5-NX I2C package are shown in the table below:

Table 2.1. Files List

Attribute	Description
soc_xo5_i2c2ifc/	This folder contains the Propel Builder project, built on the Propel 2024.1 software.
soc_xo5_i2c2ifc/impl_1/soc_xo5_i2c2ifc_impl_1.bit	This file contains the bit stream file to configure the FPGA device.
sw_soc_xo5_i2c2ifc/	This folder contains the Propel SDK project, built on the Propel 2024.1 software.
sw_soc_xo5_i2c2ifc/Debug/sw_soc_xo5_i2c2ifc.elf	This file contains the elf file to download the RISC-V program code.
rasp_pi_xo5_i2c2ifc_template.py	This file provides the host I2C commands template in programming the MachXO5-NX flash.

3. Functional Description

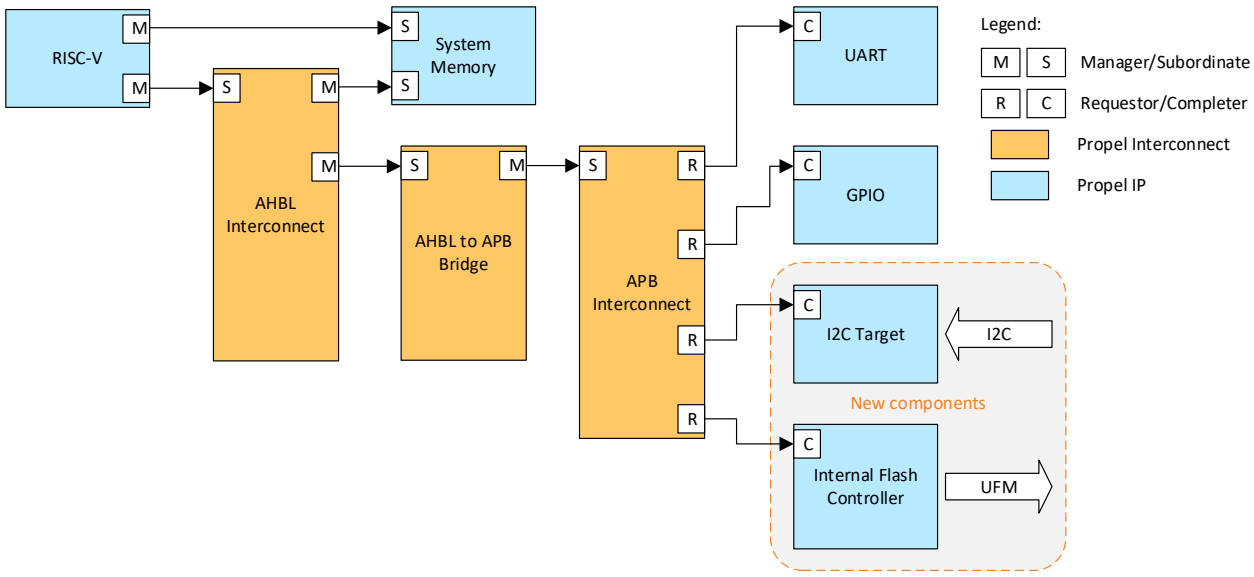

Figure 3.1. Reference Design Block Diagram

3.1. Design Components

The MachXO5-NX I2C reference design initiates and connects the MachXO5-NX flash programming through the user I2C interface. The reference design is based on the RISC-V MC SoC project template with addition of I2C Target component and Internal Flash Controller component, and removal of the Oscillator component.

The following table lists components from the Propel software that are included in the reference design. For more information on an IP, you can refer to the respective user guide.

Table 3.1. Design Components List

Soft IP Component	User Guide
RISC-V MC CPU IP	RISC-V MC CPU IP User Guide (FPGA-IPUG-02252)
System Memory Module	System Memory Module User Guide (FPGA-IPUG-02073)
GPIO IP	GPIO IP User Guide (FPGA-IPUG-02076)
UART IP	UART IP User Guide (FPGA-IPUG-02105)
I2C Target IP	I2C Target IP User Guide (FPGA-IPUG-02072)
Internal Flash Controller	MachXO5-NX Internal Flash Controller User Guide (FPGA-IPUG-02174)
PLL Module	PLL Module User Guide (FPGA-IPUG-02063)
AHB-Lite Interconnect Module	AHB-Lite Interconnect Module User Guide (FPGA-IPUG-02051)
AHB-Lite to APB Bridge Module	AHB-Lite to APB Bridge Module User Guide (FPGA-IPUG-02053)
APB Interconnect Module	APB Interconnect Module User Guide (FPGA-IPUG-02054)

Note: Each component is instantiated using the respective IP in the Lattice Propel Builder of the Propel software.

3.2. Clocking Scheme

3.2.1. Clocking Overview

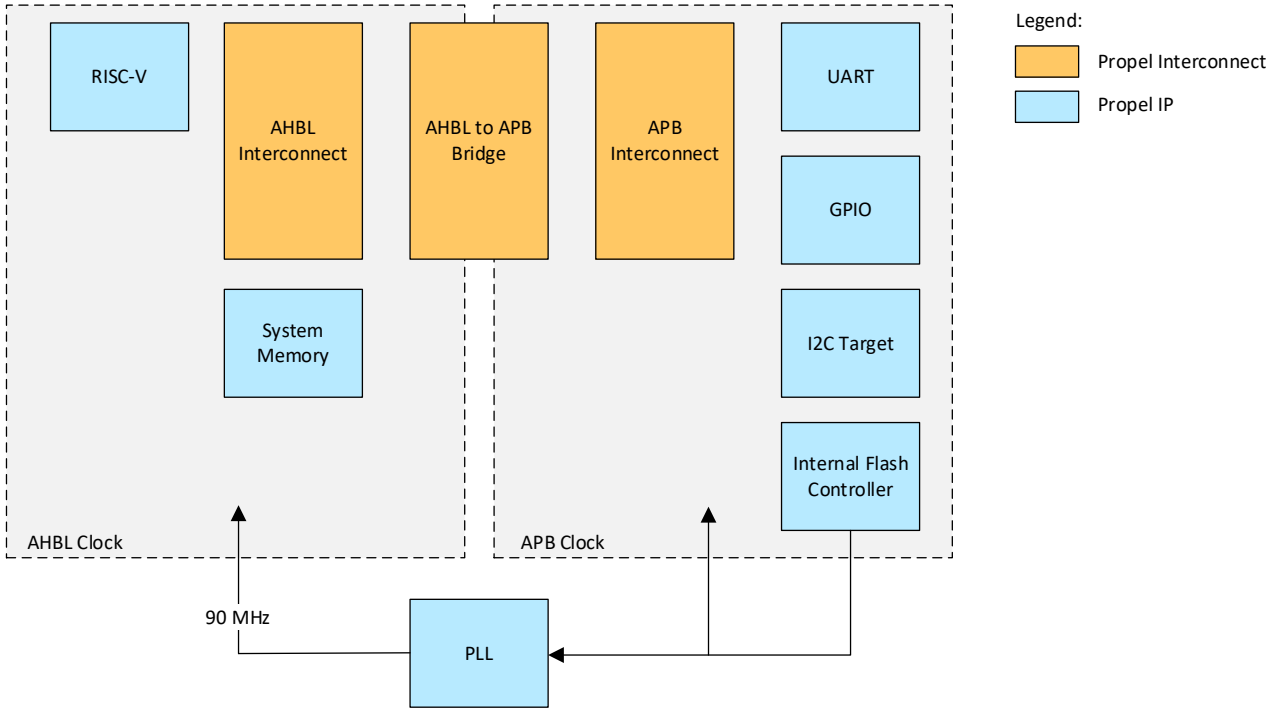

Figure 3.2. Reference Design Clock Domain Block Diagram

3.3. Reset Scheme

3.3.1. Reset Overview

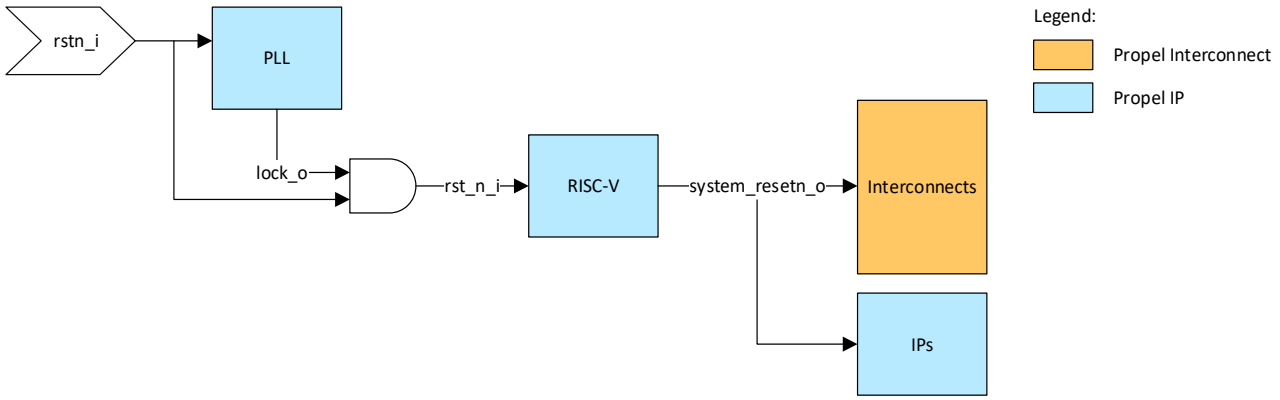

Figure 3.3. Reference Design Reset Domain Block Diagram

4. Soft-IP Blocks, Glue Logic, and GUI Configurations

4.1. RISC-V MC CPU IP

The RISC-V MC CPU IP is configured with AHBL ports for instruction and data access. Four interrupt ports are enabled for interrupt requests from the GPIO, UART, Internal Flash Controller, and I2C Target IPs.

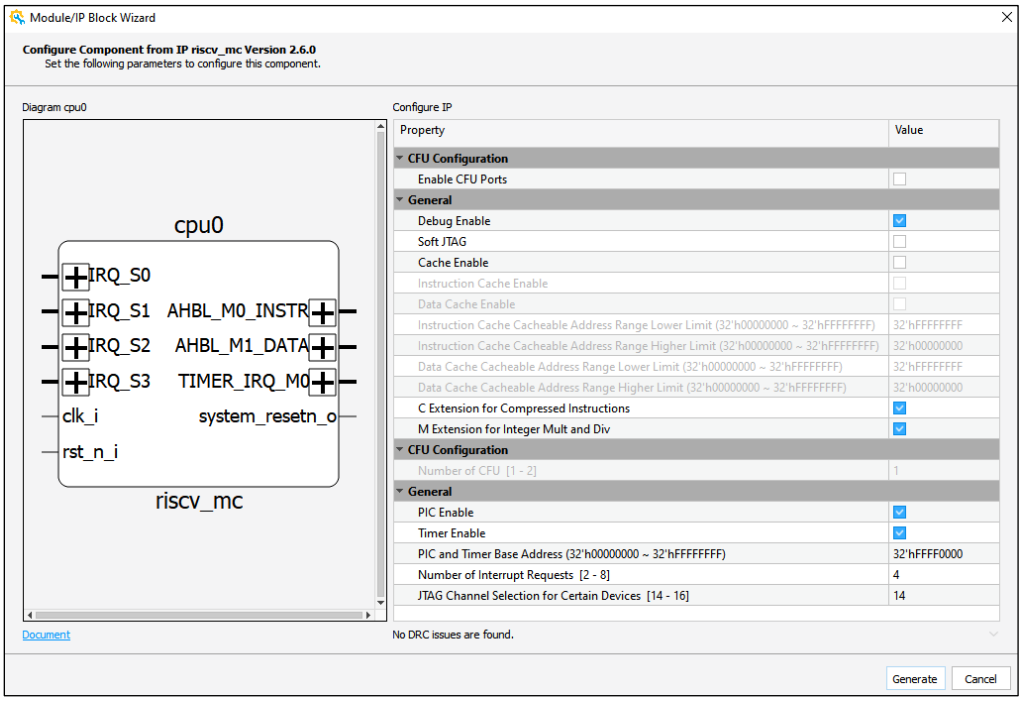

Figure 4.1. RISC-V MC CPU IP

4.2. System Memory Module

The System Memory Module is configured with dual AHBL ports. One port is for instruction memory access that connects directly to the RISC-V MC CPU IP and another port is for data memory access.

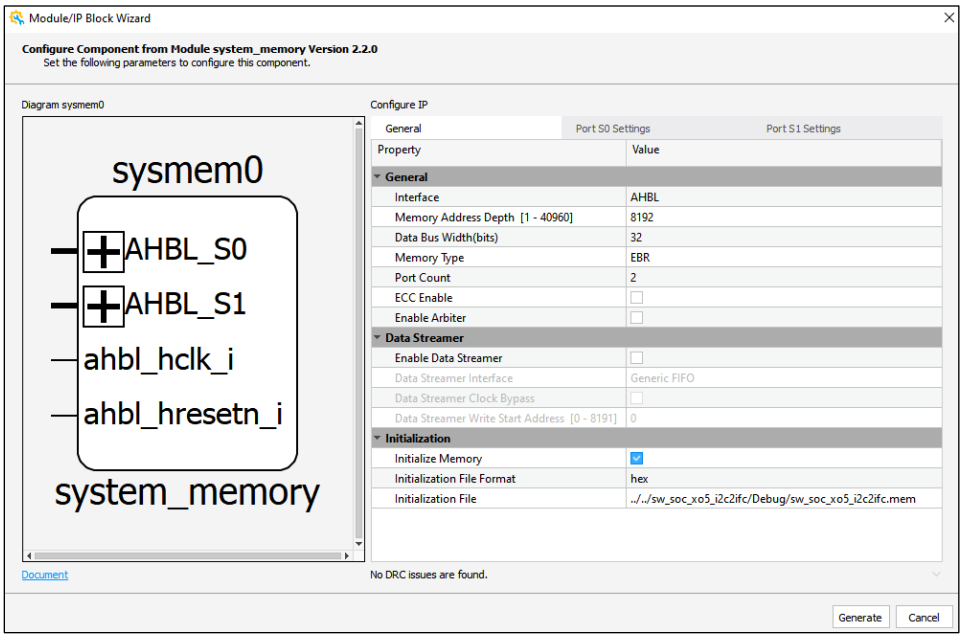

Figure 4.2. System Memory Module

4.3. GPIO IP

The GPIO IP is configured to drive the LED on the development board.

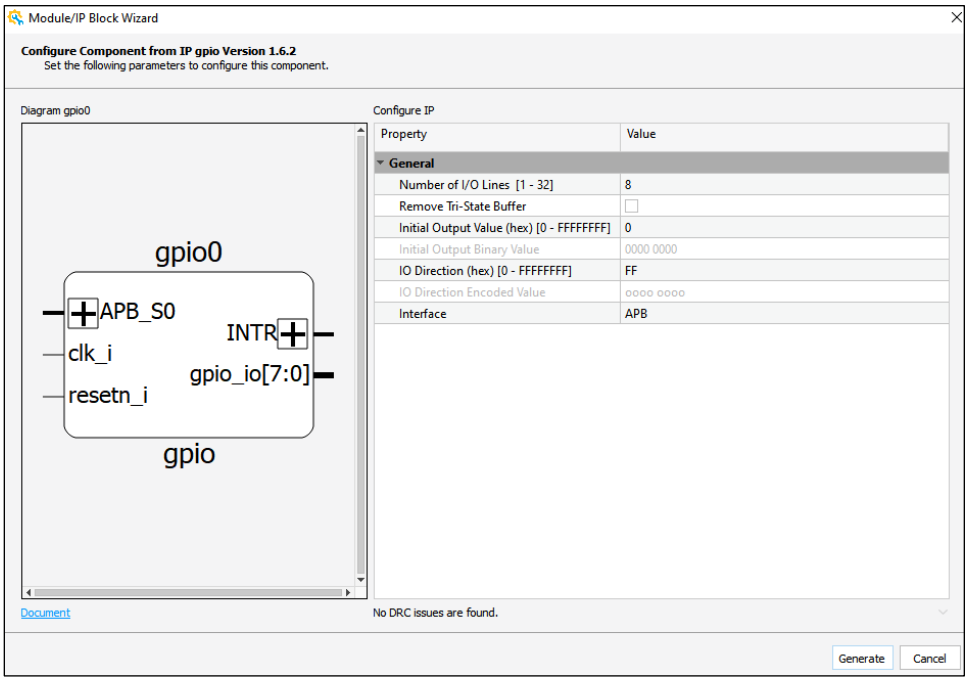

Figure 4.3. GPIO IP

4.4. UART IP

The UART IP is configured with baud rate of 115,200. The input clock is based on the APB system clock frequency.

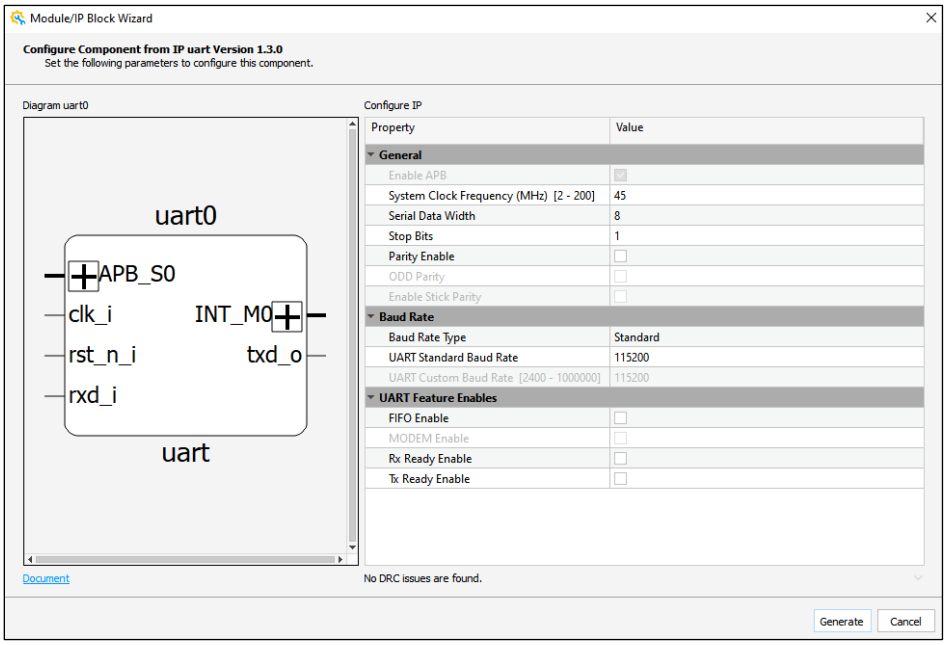

Figure 4.4. UART IP

4.5. I2C Target IP

The I2C Target IP is configured as an I2C target with the address 0x51. The FIFO depth is extended to 256 depths.

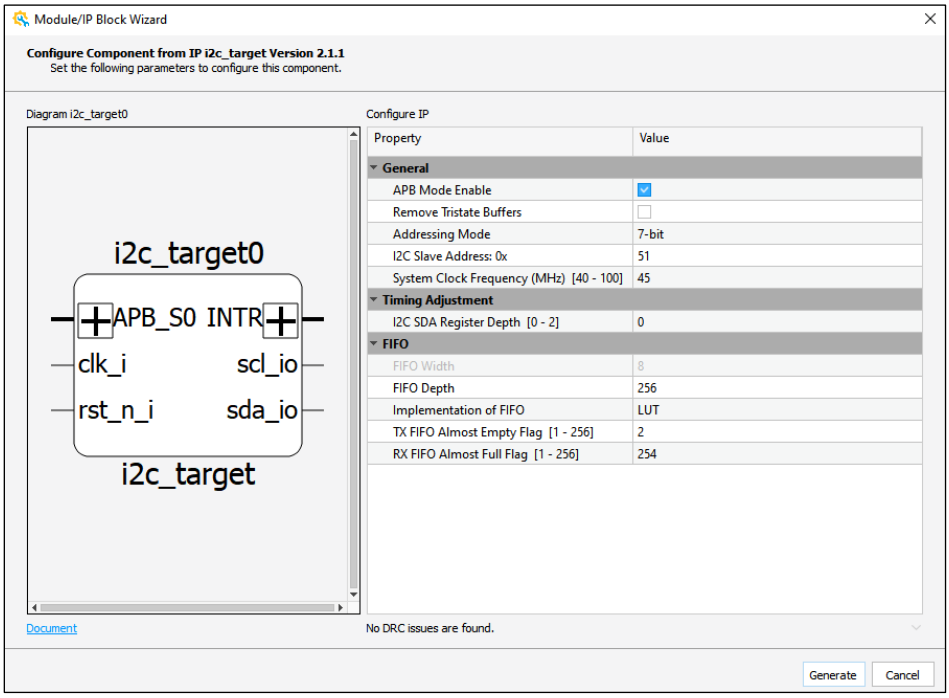

Figure 4.5. I2C Target IP

4.6. Internal Flash Controller

The Internal Flash Controller (IFC) for the MachXO5-NX devices is configured with the APB bus interface.

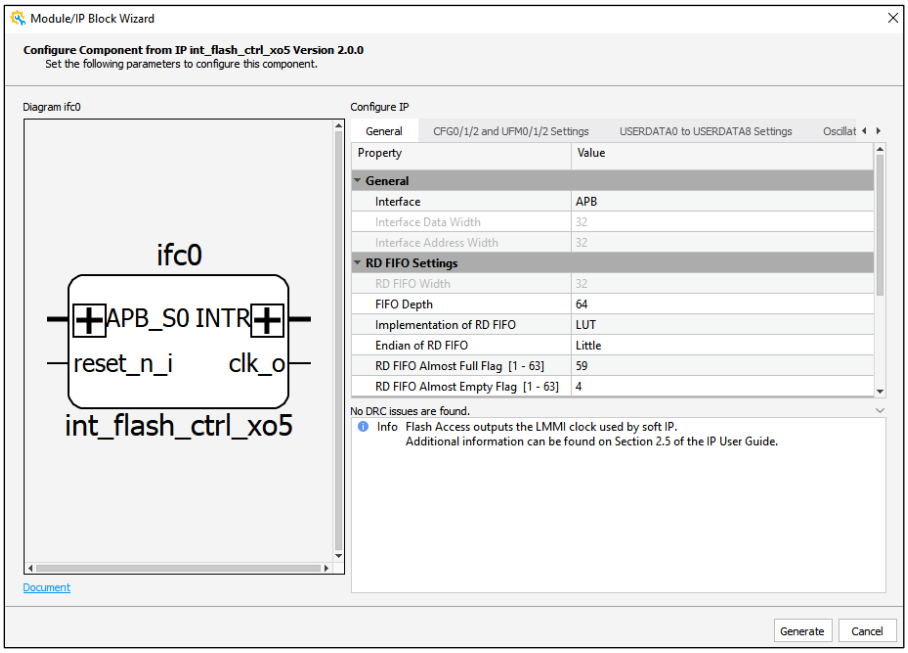

Figure 4.6. Internal Flash Controller Component

The flash partitions are enabled for CFG, UFM, and USERDATA. The block sizes for the MachXO5-NX-25 device are as follows:

- 11 blocks for CFG[0:2]
- 4 blocks for UFM[0:2]
- 2 blocks for USERDATA[0:7]
- 1 block for USERDATA[8]

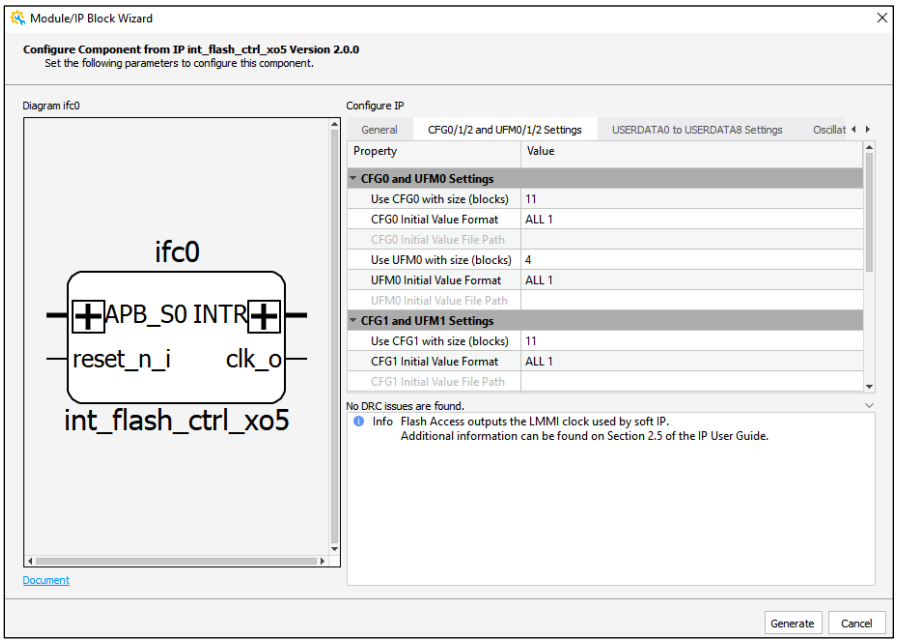

Figure 4.7. Internal Flash Controller Partition Setting

For the clock scheme, the IFC component is instantiating the internal oscillator for the LMMI and system clock source. The target frequency output is configured as 45 MHz.

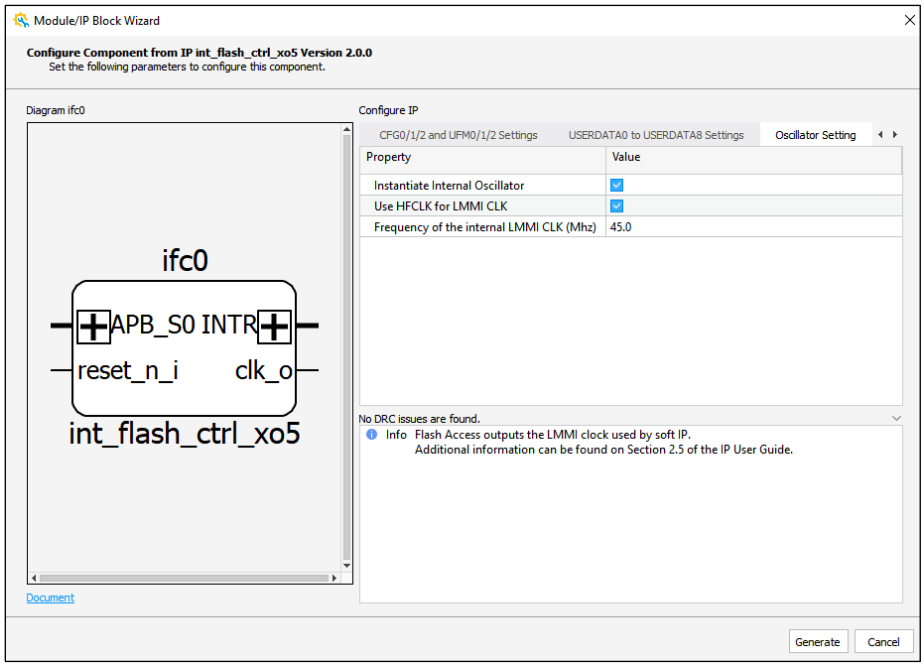

Figure 4.8. Internal Flash Controller Oscillator Setting

4.7. PLL Module

The PLL Module is configured to use the reference clock from the IFC clock output, generating a primary clock output of 90 MHz.

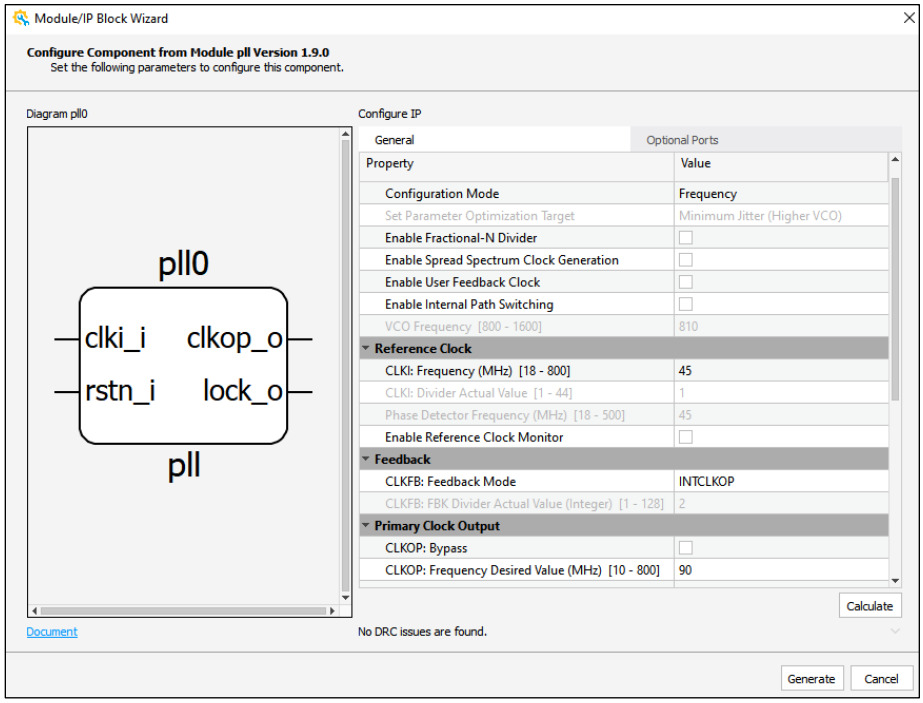

Figure 4.9. PLL Module

5. Signal Description

The I/O interface signals for the Propel SoC project are shown in the table below.

Table 5.1. Primary I/O

Port Name	I/O	Width	Description
rstn_i	Input	1	Reset input, active low
rxd_i	Input	1	UART Rx pin
txd_o	Output	1	UART Tx pin
scl_io	Input/Output	1	I2C SCL clock pin
sda_io	Input/Output	1	I2C SDA data pin
led_o	Output	8	GPIO LED output pin

6. Firmware Components

6.1. Driver Customization

The MachXO5-NX I2C reference design firmware is built on the Propel BareMetal software project with modifications on existing driver to address a hardware issue. The regenerated board support package (BSP) files are as follows:

- sw_soc_xo5_i2c2ifc\src\bsp\driver\int_flash_ctrl_xo5\ifc.h
- sw_soc_xo5_i2c2ifc\src\bsp\driver\int_flash_ctrl_xo5\ifc.c

If you need to replace any BSP file above, you can find the modified header files at the following paths:

- sw_soc_xo5_i2c2ifc\src\ifc.h_modified
- sw_soc_xo5_i2c2ifc\src\ifc.c_modified

The following figure shows the I2C data frames:

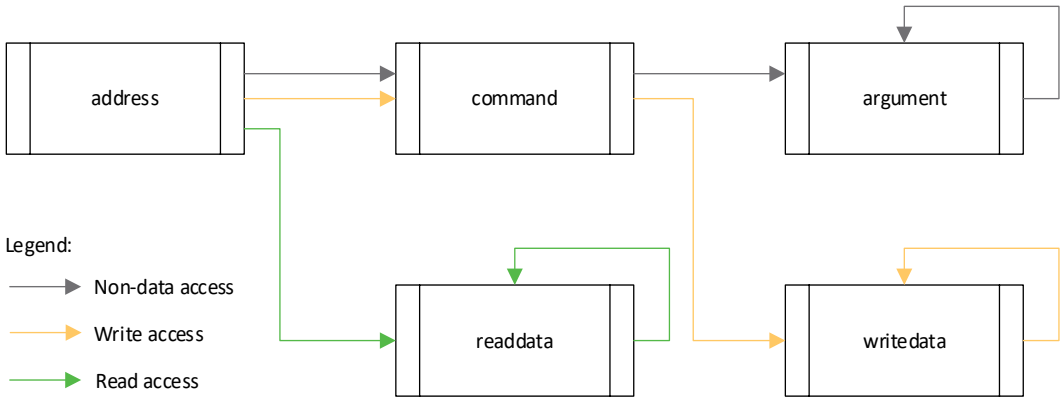

Figure 6.1. I2C Data Frames

The I2C command handling is implemented as an interrupt service routine when there is a matching incoming Rx address. All I2C commands start with first payload data as the command type, followed by operands or data. Clock stretching is enabled to prevent overflow or underflow during flash data write and read operations.

Interrupt enabled condition:

- Receiving a matching I2C address

Clock stretching conditions, which need to be cleared at the end of an interrupt service routine:

- Receiving a matching I2C address
- Transmit FIFO entering empty condition
- Receive FIFO entering full condition

The design hardware is configured with an I2C FIFO depth of 256 bytes, which accommodates the page size for write and read data. The firmware checks for Receive FIFO not empty status during the flash write operation and waits for Transmit FIFO empty status for completion of the flash read operation.

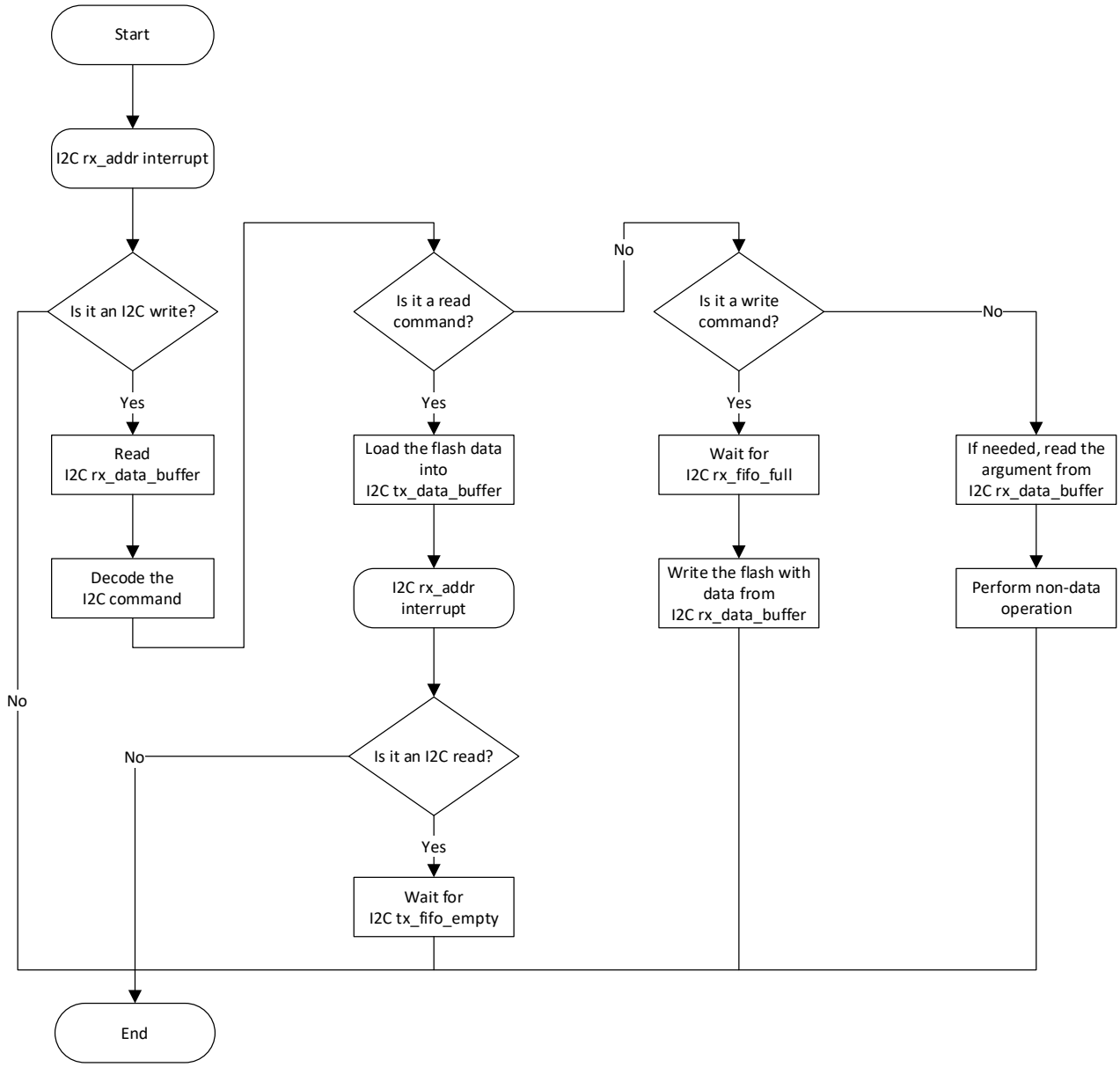

Figure 6.2. I2C to Internal Flash Data Flow

7. Host Commands

All flash data transfer between the host and internal flash is always in one-page size of 256 bytes. Operation of less than 256 bytes is not supported. Flash erasing operation is done on a per-block basis, in size of 256 pages. Multiple commands are needed to complete partition erase, write, and read operations, as the host needs to erase the destination block before writing data to the pages.

Table 7.1. Host I2C Commands

I2C command	ID	Description
XO5_CMD_SET_PAGE	0x01	Updates the current pointer of the partition and index. <I2C write> [command, partition, index[23:16], index[15:8], index[7:0]]
XO5_CMD_ERASE_FLASH	0x02	Performs block erase, based on the partition and index. <I2C write> [command, type]
XO5_CMD_CFG_RESET_ADDR	0x10	Resets the current pointer of index to 0. <I2C write> [command]
XO5_CMD_CFG_WRITE_PAGE	0x11	Performs page write of 256 bytes, then increases the pointer of index. <I2C write> [command, data * 256]
XO5_CMD_CFG_WRITE_PAGE_ALT	0x12	Performs page write of 2x128 bytes, then increases the pointer of index. <I2C write> [command, data * 128] <I2C write> [data * 128]
XO5_CMD_CFG_WRITE_PAGE_VAR	0x13	Performs page write of less than 256 bytes, then increases the pointer of index. <I2C write> [command, length] <I2C write> [data * length]
XO5_CMD_CFG_READ_PAGE	0x19	Performs page read of 256 bytes, then increases the pointer of index. <I2C write> [command] <I2C read> [data * 256]
XO5_CMD_CFG_READ_PAGE_ALT	0x1A	Performs page read of 2x128 bytes, then increases the pointer of index. <I2C write> [command] <I2C read> [data * 128] <I2C read> [data * 128]
XO5_CMD_CFG_READ_PAGE_VAR	0x1B	Performs page read of less than 256 bytes, then increases the pointer of index. <I2C write> [command, length] <I2C read> [data * length]
XO5_CMD_UFM_RESET_ADDR	0x20	Resets the current pointer of page index to 0. <I2C write> [command]
XO5_CMD_UFM_WRITE_PAGE	0x21	Performs page write of 256 bytes, then increases the pointer of page index. <I2C write> [command, data * 256]
XO5_CMD_UFM_WRITE_PAGE_ALT	0x22	Performs page write of 2x128 bytes, then increases the pointer of index. <I2C write> [command, data * 128] <I2C write> [data * 128]
XO5_CMD_UFM_WRITE_PAGE_VAR	0x23	Performs page write of less than 256 bytes, then increases the pointer of index. <I2C write> [command, length] <I2C write> [data * length]
XO5_CMD_UFM_READ_PAGE	0x29	Performs page read of 256 bytes, then increases the pointer of index. <I2C write> [command] <I2C read> [data * 256]
XO5_CMD_UFM_READ_PAGE_ALT	0x2A	Performs page read of 2x128 bytes, then increases the pointer of index. <I2C write> [command] <I2C read> [data * 128] <I2C read> [data * 128]

I2C command	ID	Description
XO5_CMD_UFM_READ_PAGE_VAR	0x2B	Performs page read of less than 256 bytes, then increases the pointer of index. <I2C write> [command, length] <I2C read> [data * length]
XO5_CMD_USERDATA_RESET_ADDR	0x30	Resets the current pointer of page index to 0. <I2C write> [command]
XO5_CMD_USERDATA_WRITE_PAGE	0x31	Performs page write of 256 bytes, then increases the pointer of page index. <I2C write> [command, data * 256]
XO5_CMD_USERDATA_WRITE_PAGE_ALT	0x32	Performs page write of 2x128 bytes, then increases the pointer of index. <I2C write> [command, data * 128] <I2C write> [data * 128]
XO5_CMD_ USERDATA _WRITE_PAGE_VAR	0x33	Performs page write of less than 256 bytes, then increases the pointer of index. <I2C write> [command, length] <I2C write> [data * length]
XO5_CMD_USERDATA_READ_PAGE	0x39	Performs page read of 256 bytes, then increases the pointer of index. <I2C write> [command] <I2C read> [data * 256]
XO5_CMD_USERDATA_READ_PAGE_ALT	0x3A	Performs page read of 2x128 bytes, then increases the pointer of index. <I2C write> [command] <I2C read> [data * 128] <I2C read> [data * 128]
XO5_CMD_USERDATA_READ_PAGE_VAR	0x3B	Performs page read of less than 256 bytes, then increases the pointer of index. <I2C write> [command, length] <I2C read> [data * length]

Follow the following steps to perform block erase:

1. Send the XO5_CMD_SET_PAGE command with targeted partition number and block index.
2. Send the XO5_CMD_ERASE_FLASH command.

Follow the following steps to perform page write:

1. Send the XO5_CMD_SET_PAGE command with targeted partition number and page index.
2. Send the XO5_CMD_*_WRITE_PAGE command, followed by 256 bytes of write data.
3. Repeat step 2 for subsequent write data page until completion.

Follow the following steps to perform page read:

1. Send the XO5_CMD_SET_PAGE command with targeted partition number and page index.
2. Send the XO5_CMD_*_READ_PAGE command.
3. Read 256 bytes of read data.
4. Repeat step 2 and 3 for subsequent read data page until completion.

To reset the page index back to 0, send the XO5_CMD_*_ RESET_ADDR command.

To execute the Python script, run *python rasp_pi_xo5_i2c2ifc_template.py* in a Linux console such as the Raspberry Pi terminal. The template performs regression of write and read operations on the CFG [1:2], UFM [0:2], and USERDATA [0:8] partitions.

Below is an example of writing to UFM0 (partition index of 0x1), which consist of 4 blocks:

1. Send the I2C bytes of [0x01, 0x01, 0x00, 0x00, 0x00] for XO5_CMD_SET_PAGE.
2. Send the I2C bytes of [0x02] for XO5_CMD_ERASE_FLASH.
3. Send the I2C bytes of [0x20, 256 write bytes] for XO5_CMD_UFM_WRITE_PAGE.
4. Repeat step 3 until all write bytes for the block have been written.
5. Repeat steps 2, 3, and 4 for each subsequent block until all blocks have been written.
6. Send the I2C bytes of [0x22] for XO5_CMD_UFM_RESET_ADDR to reset the index back to 0.
7. Send the I2C bytes of [0x21] for XO5_CMD_UFM_READ_PAGE, then read the 256 I2C read bytes.
8. Repeat step 7 until all blocks have been read.

8. Design Constraints

8.1. Clock Constraints

The clock constraints are automatically provided by the components and are reported as follows:

```
create_clock -name
{ifc0_inst/ufm_access_inst/secured_instance_81_10/secured_instance_80_1/secured_signal_79_6} -period
4.44444 [get_pins
{ifc0_inst/ufm_access_inst/secured_instance_81_10/secured_instance_80_1/secured_instance_79_1/HFSDCOUT
}]
create_clock -name {ifc0_inst_clk_o_net} -period 22.2222 [get_pins
{ifc0_inst/ufm_access_inst/secured_instance_81_10/secured_instance_80_1/secured_instance_79_1/HFCLKOUT
}]
create_generated_clock -name {pll0_inst_clkop_o_net} -source [get_pins
{pll0_inst/lscc_pll_inst/gen_no_refclk_mon.u_PLL.PLL_inst/REFCK}] -multiply_by 2 [get_pins
{pll0_inst/lscc_pll_inst/gen_no_refclk_mon.u_PLL.PLL_inst/CLKOP }]
```

8.2. I/O Constraints

The pins assignments are manually defined in the Post-Synthesis Physical (PDC) constraints as follows:

```
ldc_set_location -site {F20} [get_ports rstn_i]
ldc_set_location -site {B11} [get_ports rxd_i]
ldc_set_location -site {B12} [get_ports txd_o]
ldc_set_location -site {D5} [get_ports scl_io]
ldc_set_location -site {B5} [get_ports sda_io]
ldc_set_location -site {R3} [get_ports {led_o[0]}]
ldc_set_location -site {R2} [get_ports {led_o[1]}]
ldc_set_location -site {R1} [get_ports {led_o[2]}]
ldc_set_location -site {P7} [get_ports {led_o[3]}]
ldc_set_location -site {H12} [get_ports {led_o[4]}]
ldc_set_location -site {H11} [get_ports {led_o[5]}]
ldc_set_location -site {G13} [get_ports {led_o[6]}]
ldc_set_location -site {G12} [get_ports {led_o[7]}]

ldc_set_port -iobuf {IO_TYPE=LVCMOS33} [get_ports rxd_i]
ldc_set_port -iobuf {IO_TYPE=LVCMOS33} [get_ports txd_o]
```

9. Running the Reference Design

This section describes how to run the reference design using the Lattice Radiant software. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software user guide.

9.1. Compiling the Reference Design

9.2. Generating the Bitstream File

This section provides the procedure of creating your FPGA bitstream file using the Lattice Radiant software. To create the FPGA bitstream file, follow the steps below.

1. Open the Lattice Radiant software.

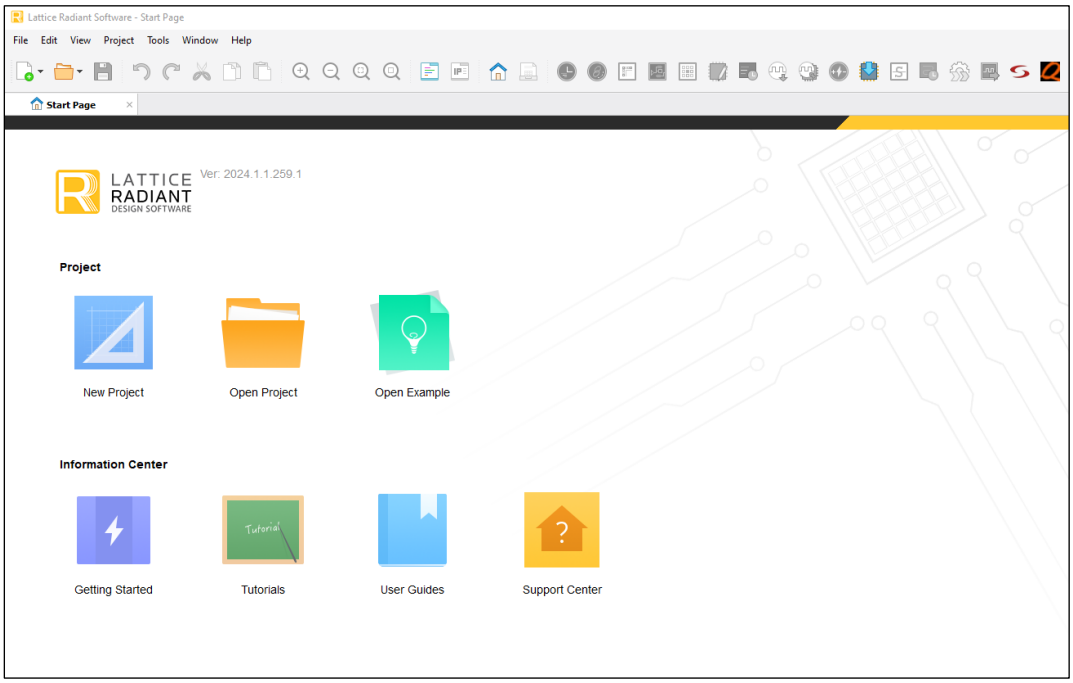

Figure 9.1. Lattice Radiant Software

2. Click **File** > **Open Project** from the project database and open the Radiant project file (.rdf) from the **<FOLDER NAME>** folder.

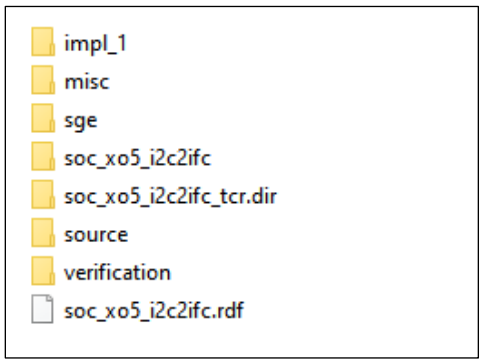

Figure 9.2. Open Project File

3. Click **Export Files** to generate the bit file. View the log message in the **Export Reports** folder for the generated bitstream as shown below. For the Radiant software version 2024.1, the *1009990 ERROR* message is expected to appear as a warning message.

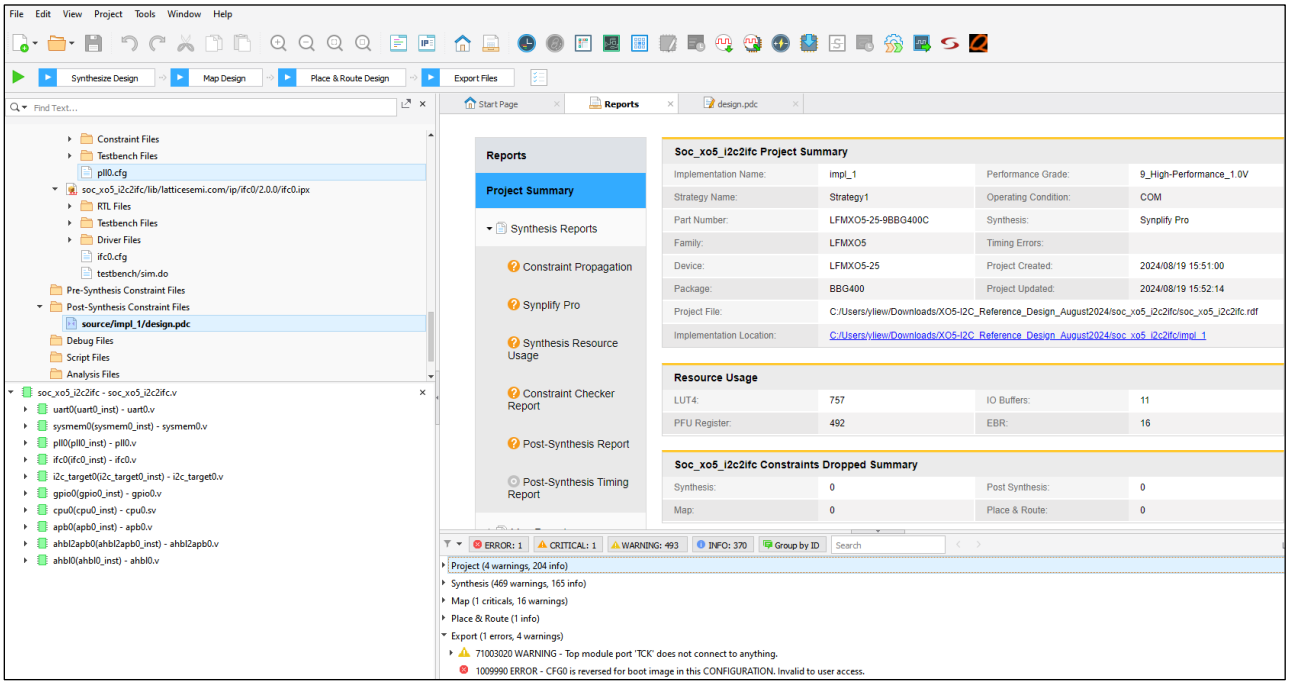

Figure 9.3. Generated Bitstream Log

10. Simulating the Reference Design

To simulate the reference design, perform the following steps:

1. Unzip the reference design zip file.
2. Open the reference design Propel project file (soc_xo5_i2c2ifc.sbx) with the Lattice Propel software.

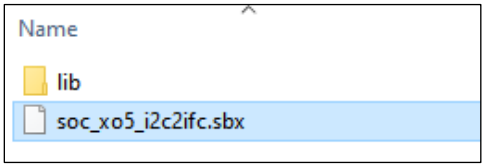

Figure 10.1. Propel Project File

3. Click on **Design** > **Switch Verification and SoC Design**.

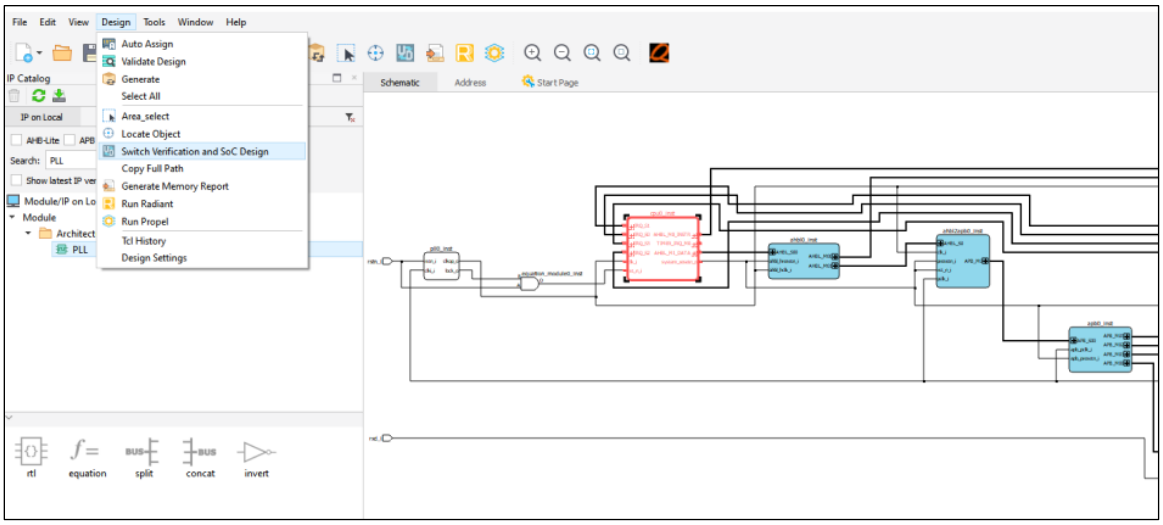

Figure 10.2. Propel Project Design View

4. Click on **Tools** > **Questasim Lattice-Edition** and wait for the QuestaSim Lattice-Edition software to load.

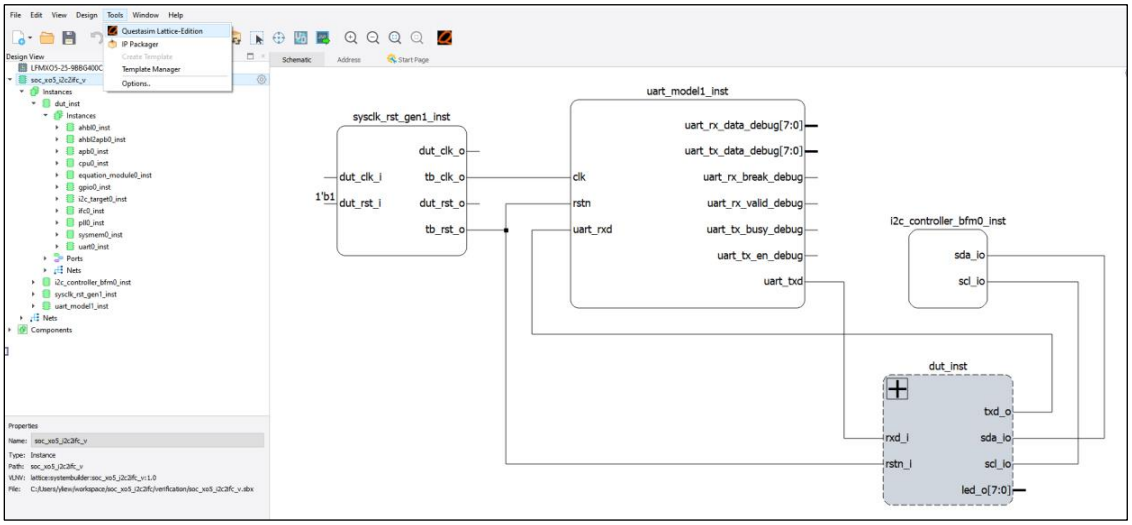

Figure 10.3. Propel Project Verification View

5. To load the .do file, click on **Tools** > **TCL** > **Execute Macro**, and select the socsim.do file from the provided zip file.

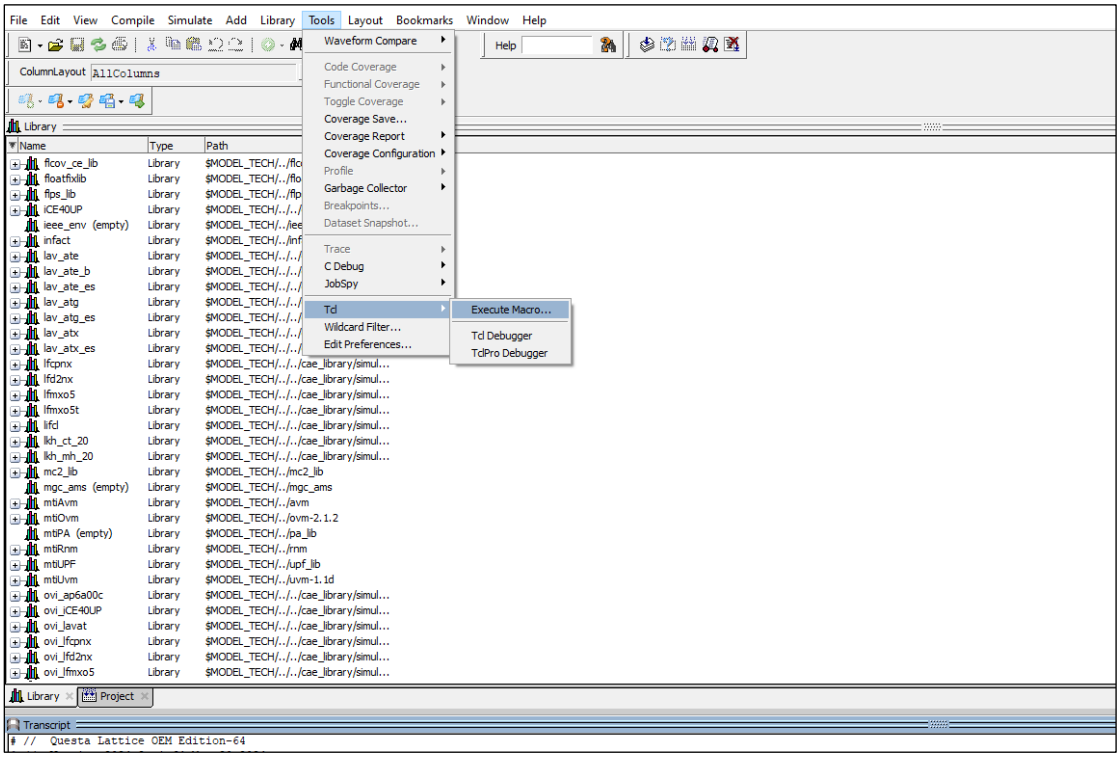

Figure 10.4. Execute socsim.do File in the QuestaSim Interface

6. The simulation starts.

10.1. Simulation Result

The simulation wave is shown below. Note that the erase operation might take up to 200 ms to complete.

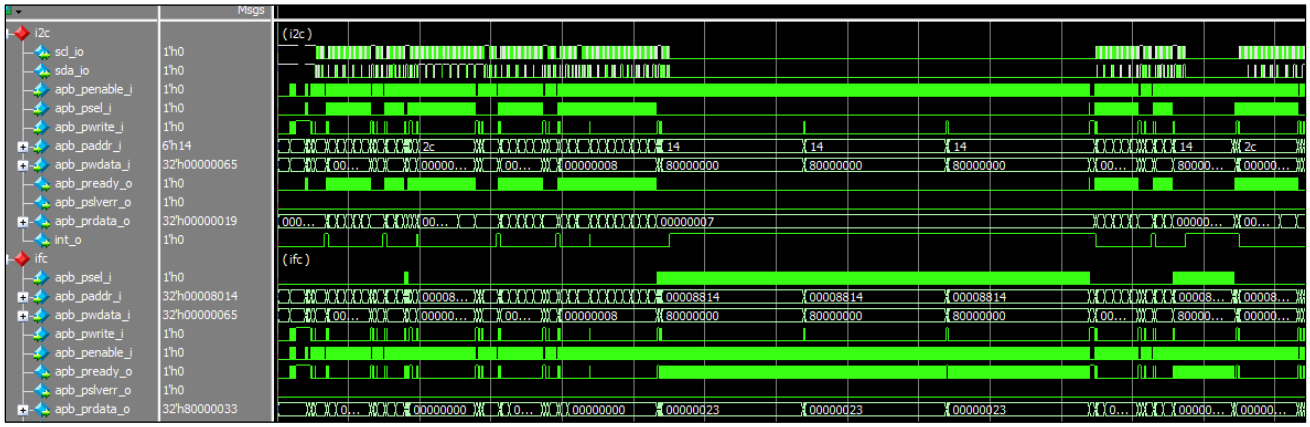

Figure 10.5. Waveform Signals in the QuestaSim Interface

11. Implementing the Reference Design on Board

The MachXO5-NX development board shown below is used for the reference design.

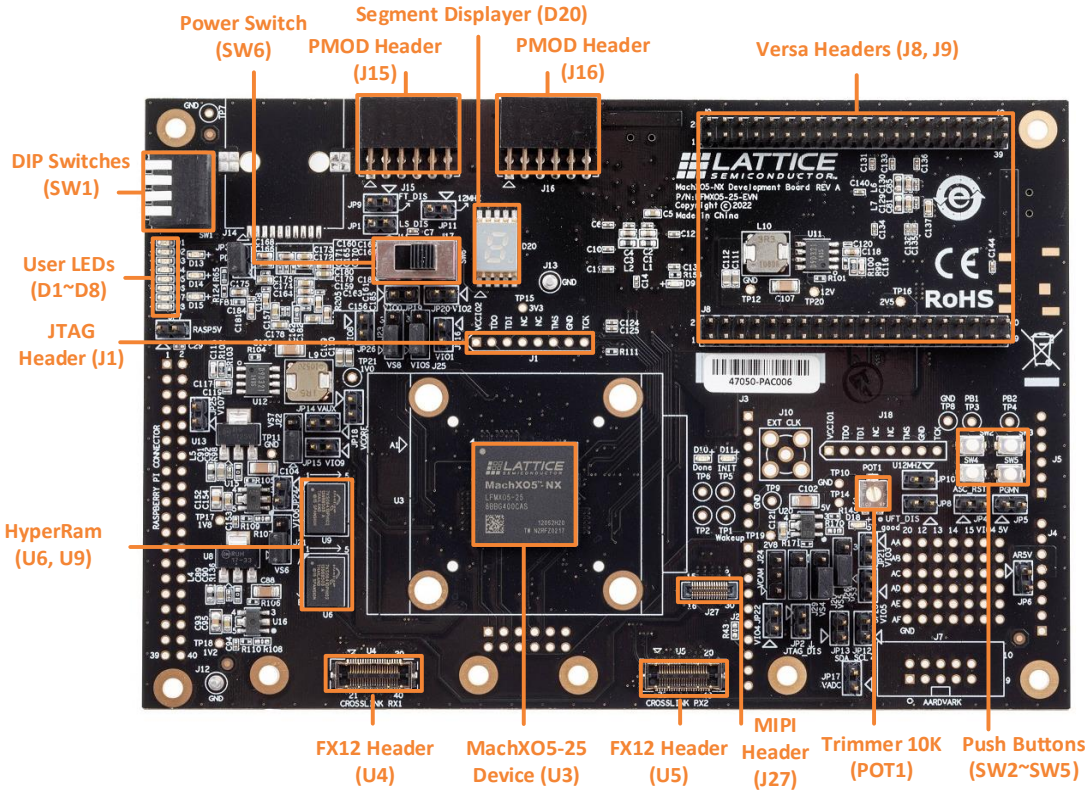

Figure 11.1. MachXO5-NX Development Board

To implement the reference design on the development board, perform the following steps:

1. From the Radiant project, select **Tools** > **Programmer**.

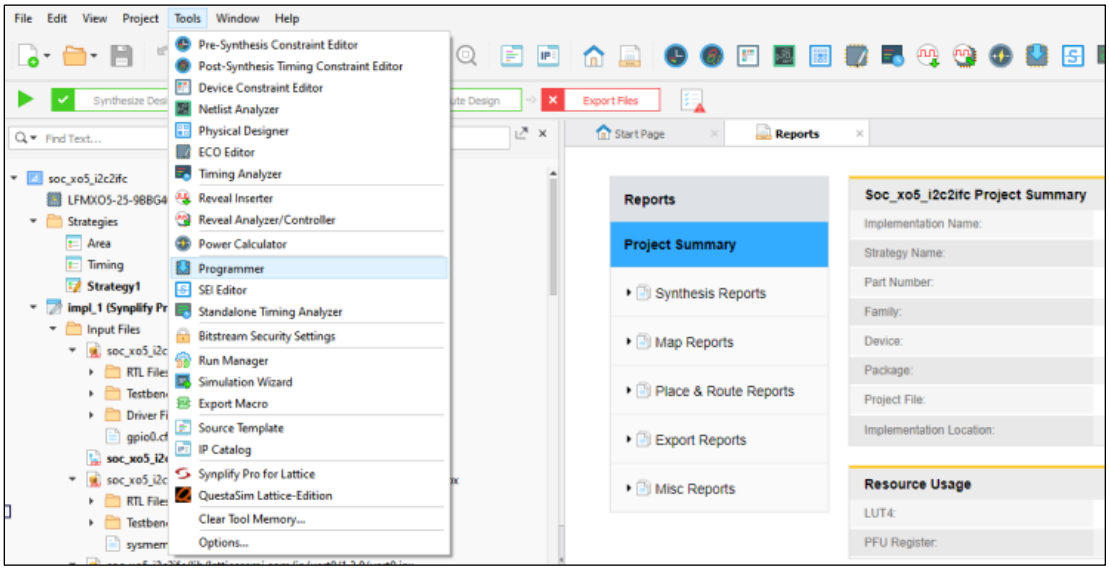

Figure 11.2. Lattice Radiant Software GUI

2. From the Programmer tool, select **Run** > **Program Device** to start programming the FPGA device.

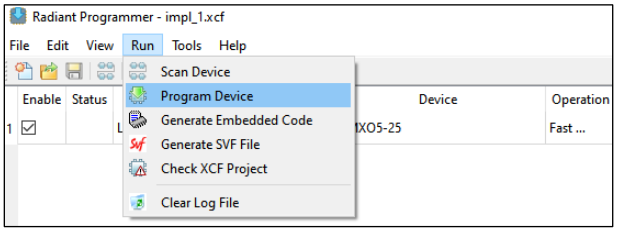

Figure 11.3. Lattice Radiant Programmer GUI

12. Resource Utilization

The resource utilization summary is provided in the Place and Routing section of the Radiant software report.

Device utilization summary:

VHI	1/1	100% used
SIOLOGIC	9/306	3% used
DCC	1/62	2% used
EBR	22/80	28% used
MULT9	12/40	30% used
MULT18	6/20	30% used
MULT18X36	2/10	20% used
REG18	10/40	25% used
PREADD9	12/40	30% used
SEIO33	13/299	4% used
	13/251	5% bonded
OSC	1/1	100% used
PLL	1/2	50% used
CONFIG_JTAG	1/1	100% used
CONFIG_LMMIB	1/1	100% used
CONFIG_CLKRST	1/1	100% used
SLICE	7217/11520	63% used
LUT	6093/23040	26% used
REG	3394/23040	15% used

Figure 12.1. Device Utilization Summary

13. Known Limitations

The know limitations of the MachXO5-NX I2C reference design are as follows:

- The reference design has only verified with hardware between the FPGA device and Raspberry Pi on limited testing. No simulation testing is performed.
- The CFG0 partition enablement generates specific error messages during the bitstream generation at the export stage. The bit file is generated and can be programmed into the FPGA device.
- The python script is provided as a template for the I2C command flow, serving as a reference for porting to an embedded C implementation.
- The I2C driver has been modified as a workaround for a hardware issue, and needs to be reverted once the actual hardware issue is resolved.

References

- RISC-V MC CPU IP User Guide (FPGA-IPUG-02252)
- System Memory Module User Guide (FPGA-IPUG-02073)
- GPIO IP User Guide (FPGA-IPUG-02076)
- UART IP User Guide (FPGA-IPUG-02105)
- I2C Target IP User Guide (FPGA-IPUG-02072)
- MachXO5-NX Internal Flash Controller User Guide (FPGA-IPUG-02174)
- PLL Module User Guide (FPGA-IPUG-02063)
- AHB-Lite Interconnect Module User Guide (FPGA-IPUG-02051)
- AHB-Lite to APB Bridge Module User Guide (FPGA-IPUG-02053)
- APB Interconnect Module User Guide (FPGA-IPUG-02054)
- Lattice Radiant Software 2024.1 User Guide
- MachXO5-NX web page
- Lattice Solutions Reference Designs web page
- MachXO5-NX Development Board web page
- RISC-V MC CPU IP Core web page
- System Memory Module IP Core web page
- GPIO IP Core web page
- I2C Target IP Core web page
- MachXO5-NX Internal Flash Controller IP Core web page
- AHB-Lite Interconnect Module web page
- AHB-Lite to APB Bridge Module web page
- APB Interconnect Module web page
- Lattice Solutions IP Cores web page
- Lattice Solutions Reference Designs web page
- Lattice Propel Design Environment web page
- Lattice Radiant Software web page
- Lattice Insights web page for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

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

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

Revision History

Revision 1.0, January 2025

Section	Change Summary
All	Initial release.

www.latticesemi.com