MPESTI Initiator

Reference Design

FPGA-RD-02274-1.2

October 2023

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and 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 REDUDANCIES, 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

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AHB	Advanced High-performance Bus
APB	Advanced Peripheral Bus
EBR	Embedded Block RAM
FIFO	First In, First Out
FPGA	Field Programmable Gate Array
JTAG	Joint Test Action Group
LED	Light-Emitting Diode
LUT	Look-Up Table
RAM	Random Access Memory
ROM	Read-Only Memory
UART	Universal Asynchronous Receiver-Transmitter
VW	Virtual Wire

1. Introduction

The MPESTI Initiator Reference Design provides a solution template that uses the MPESTI Initiator IP core, Initiator pattern generator block, and MPESTI Target test component. The Reference Design is compliant to the MPESTI Base Specification (part of the DC-MHS version 1.0 specification).

1.1. Features

The key features of the MPESTI Initiator include:

- MPESTI Initiator IP core and MPESTI Target test component communicate via half-duplex bidirectional UART protocol at 250k BAUD rate, 8-bit data, 1-bit odd parity, 1 start bit and 1 stop bit (MPESTI line)
- Supports Static Discovery payload with CRC-8 payload checksum
- Supports one-manager-to-many-subordinates mode
- Supports APB collector interface (subordinate) to communicate with DC-SCM LTPI IP core and transfer data to DC-SCM module
- Supports configurable number of MPESTI line
- Supports broadcast power break command to all connecting MPESTI Targets
- Supports auto trigger Static Discovery Payload request command to all Targets on round robin manner during Discovery phase
- Supports Static Discovery Payload request command retry. If payload is not successfully received after an initial attempt, two more retries per target are triggered
- Supports Target reset at any time and fault handling such as RX timeout

1.2. Limitations

The MPESTI Initiator IP core supports MachXO5™-NX, MachXO3D™, MachXO3L™, MachXO3LF™ and Avant™ devices.

2. Functional Description

2.1. Overview

Figure 2.1 shows the high-level block diagram of the reference design.

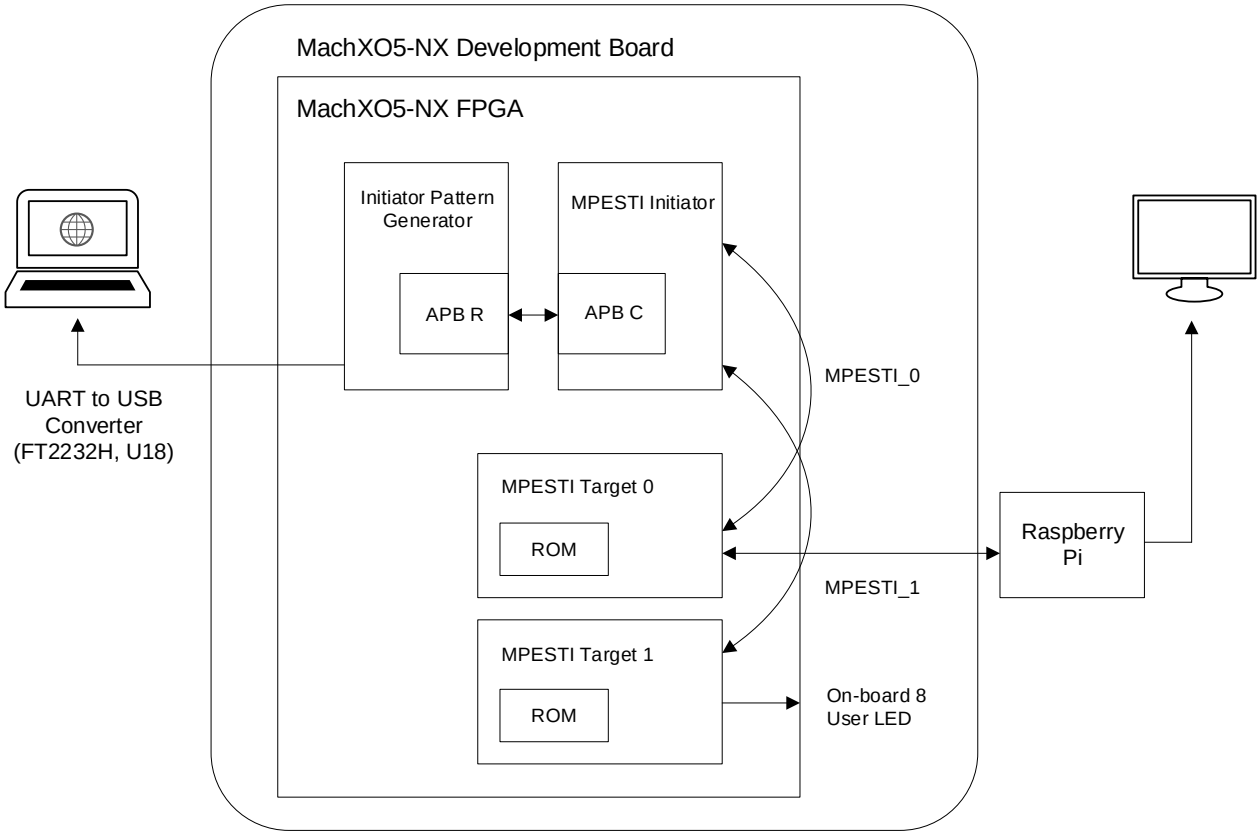


Figure 2.1. Functional Block Diagram

The Pattern Generator is created to drive APB transactions to MPESTI Initiator core, and includes the following functions:

- Set up the registers of the Initiator core during initialization state.
- Read out data from Static Discovery Payload RAM in the Initiator IP and convert them into UART transaction, then display them on PC monitor using ComDebug Serial Terminal Application.
- Includes a counter, which counts from 0 to NUM_VIRTUAL_WIRE_OUTPUT_BYTES[3:0]. This counter value is used to indicate the Virtual Wire (VW) output data bytes that is written into VW OUT FIFO register in the Initiator core.
- Read out response data from VW IN FIFO register in the Initiator core and convert them into UART transaction, then display them on the PC monitor using ComDebug Serial Terminal Application.

The MPESTI Target IP is created to return the Discovery/VW response payloads to the MPESTI Initiator. There is a ROM block instantiated inside the Target, which is initialized with Discovery Payload data bytes.

MPESTI Target 0's 8-bit VW input/output ports connects to Raspberry Pi minicomputer through GPIO pins. A simple GUI is set up for user to send 8-bit VW response byte to MPESTI Initiator (such as 8'h00, 8'h0A, and 8'h0B). The VW data bytes received by the MPESTI Target from the Initiator (counter value) are displayed on the screen. Figure 2.2 and Figure 2.3 shows the user interface of the Raspberry Pi minicomputer.

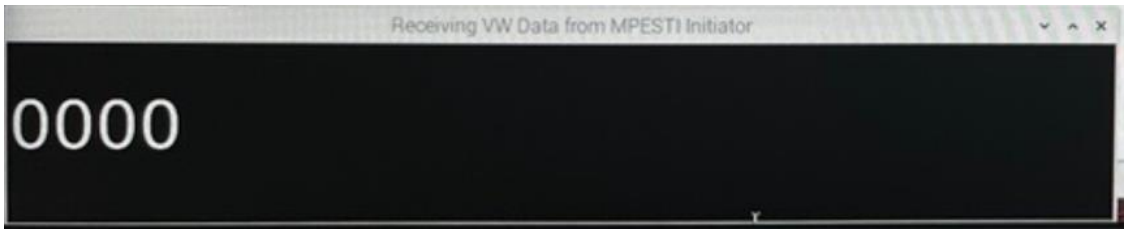


Figure 2.2. Raspberry Pi Mini Computer Receiving Data User Interface

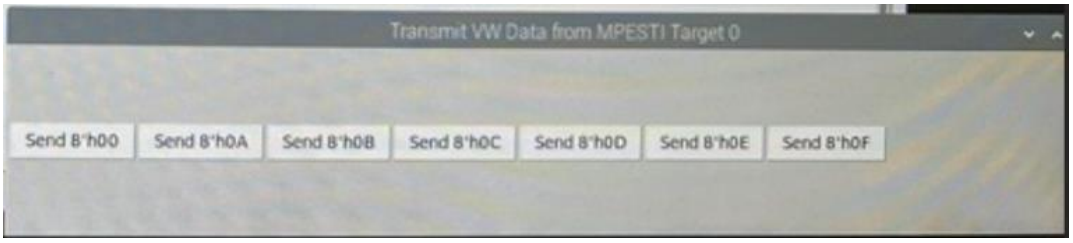


Figure 2.3. Raspberry Pi Mini Computer Transmitting Data User Interface

The MPESTI Target 1's 8-bit VW output ports connects to eight user LEDs on the development board. After Target 1 has received the 8-bit VW data bytes from the Initiator, it then sends the 8-bit VW response bytes back to Target 1. These response bytes are formed by concatenating 4-bit <Target Index number> and 4-bit <X>, where X – counts up from 0 to NUM_VW_IN_BYTE.

To configure the ComDebug serial terminal application, perform the steps below:

1. Open ComDebug and select Use the program as a Terminal Utility.

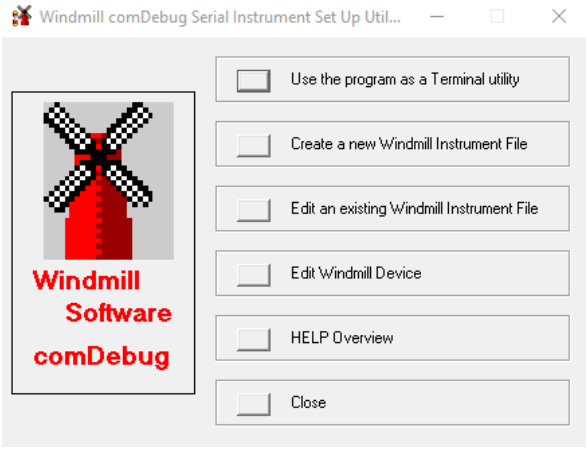


Figure 2.4. Windmill ComDebug Serial Instrument Set-up

2. Click **Serial Communications**.

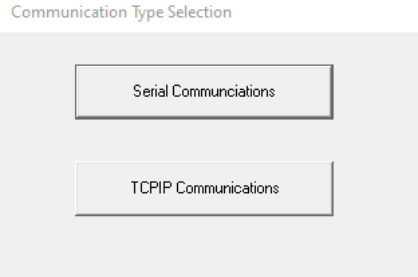


Figure 2.5. Communication Type Selection

3. Select the **UART Comm** for the port settings. Set Baud Rate to **115200**, Data Bits to **8**, Parity to **Odd**, Stop bits to **1**, and Flow Control to **None**. Click **OK**.

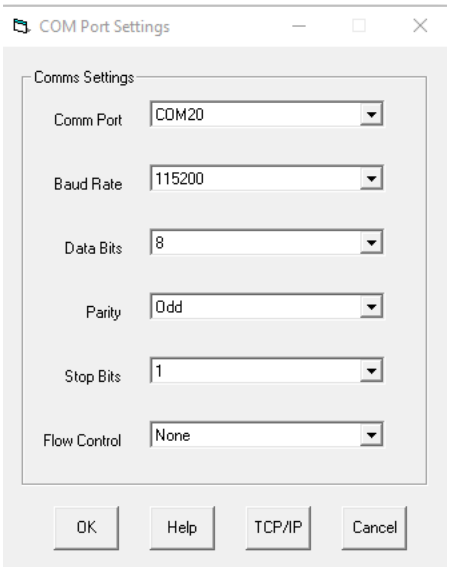


Figure 2.6. COM Port Settings

4. The ComDebug terminal screen displays the received UART data bytes as shown in Figure 2.7.

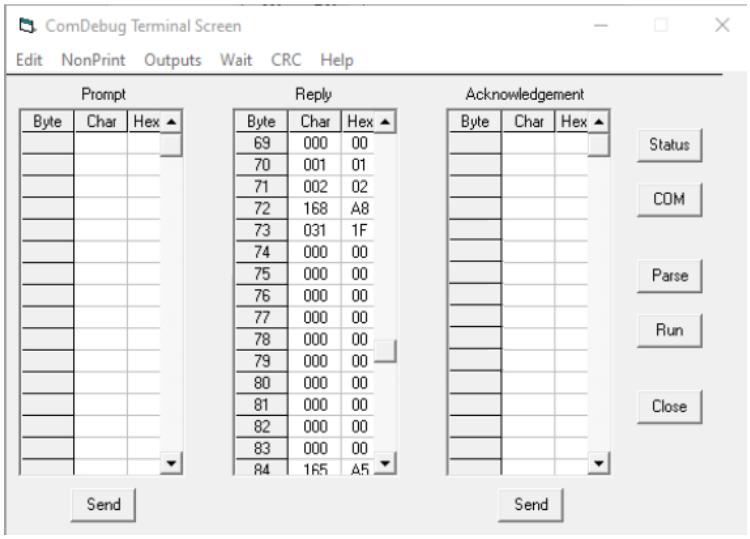


Figure 2.7. ComDebug Terminal Screen

2.2. Detailed Description

2.2.1. Discovery Phase

After power on, the MPESTI Target is in initialization state (break event) that lasts for T_DBREAK_NS (50 µs). After that, the MPESTI Target releases the MPESTI line (break release) and gets ready to respond to the Discovery Payload request command. Figure 2.8 shows the Discovery Phase. The first 8-bit transaction asserted during tx_in_progress is the Discovery payload request command. The next 8-bit transaction during rcv_in_progress is the Discovery Payload data.

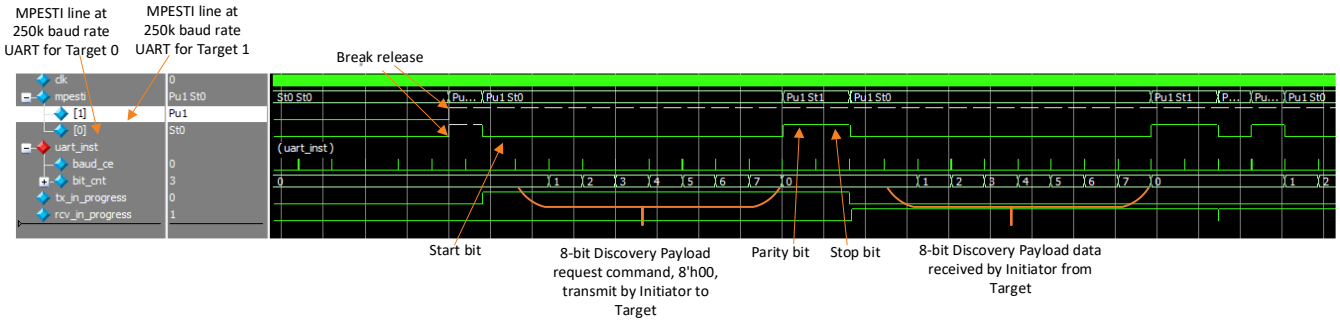


Figure 2.8. Discovery Phase

Target 0 responds with 16 Discovery Payload data bytes which matches the third header byte value (STATIC_PAYLOAD_SIZE[7:0] x 8) of the Discovery Payloads. According to MPESTI spec, the value of STATIC_PAYLOAD_SIZE represents the number of [Discovery Payload data bytes]/8. Example: STATIC_PAYLOAD_SIZE = 02h indicates the size as 2 × 8 = 16 discovery payload data bytes.

Bit[3:0] of 4th byte indicates the number of VW input bytes from Target to Initiator (NUM_VW_IN_BYTE).

Bit[7:4] of 4th byte indicates the number of VW output bytes from Initiator to Target (NUM_VW_OUT_BYTE).

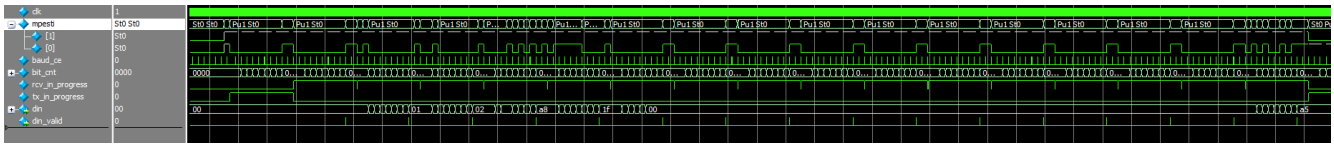


Figure 2.9. 16 Discovery Pay Load Data Bytes

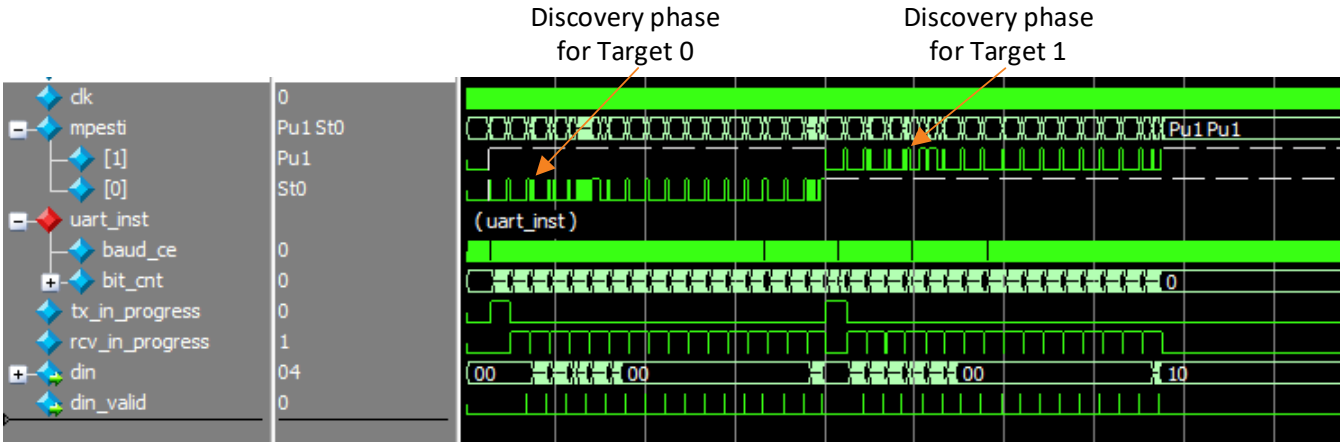


Figure 2.10. Round Robin Discovery Phase on Target 0 and Target 1

2.2.2. Active Phase

During the active phase, the Initiator issues a VW exchange request command to the Target, 8'h01 and starts VW exchange once VWOUT FIFO is not empty. The length of the VWOUT data bytes depends on the NUM_VW_OUT_BYTE. Next, the Target responds with VW payload based on/matching the value of NUM_VW_IN_BYTE. Figure 2.11 shows the VW data exchange activity.

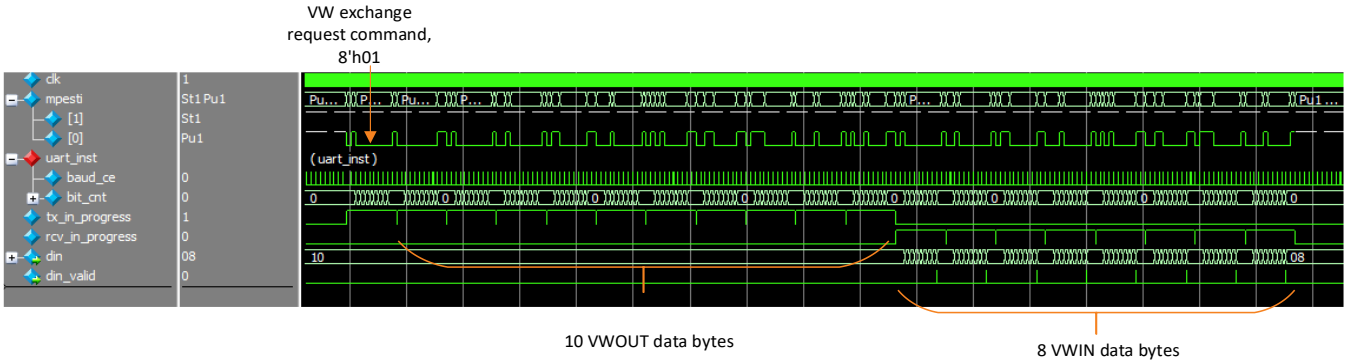


Figure 2.11. Active Phase Virtual Wire Exchange Data

2.2.3. Broadcast Transaction

The broadcast request can only be triggered during the active phase. However, if the broadcast request is triggered when VW exchange is in progress, it is delayed until the VW exchange completes. This scenario is shown in Figure 2.12, where the broadcast request command ('hFF) only happens at the MPESTI line after the VW exchange completes.

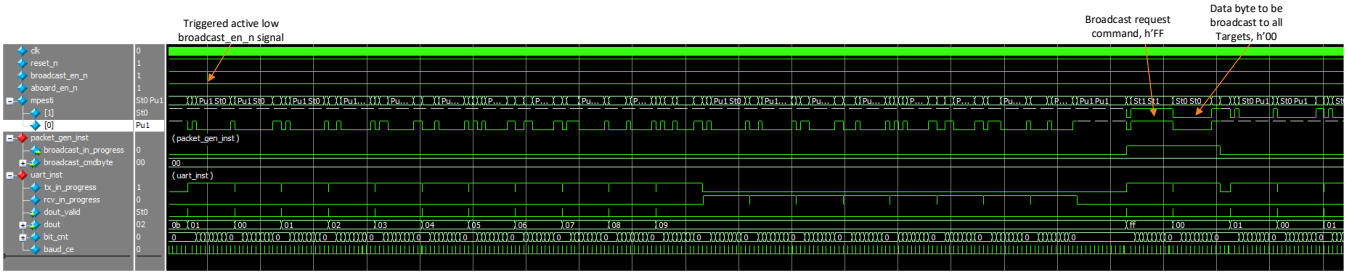


Figure 2.12. Broadcast Request

2.2.4. Abort transaction

The Abort Assertion can only be triggered during active phase. The Initiator abort assertion (tABREAK) occurs during the following:

- The Initiator is transmitting or about to transmit.
- The Target is transmitting or about to transmit.

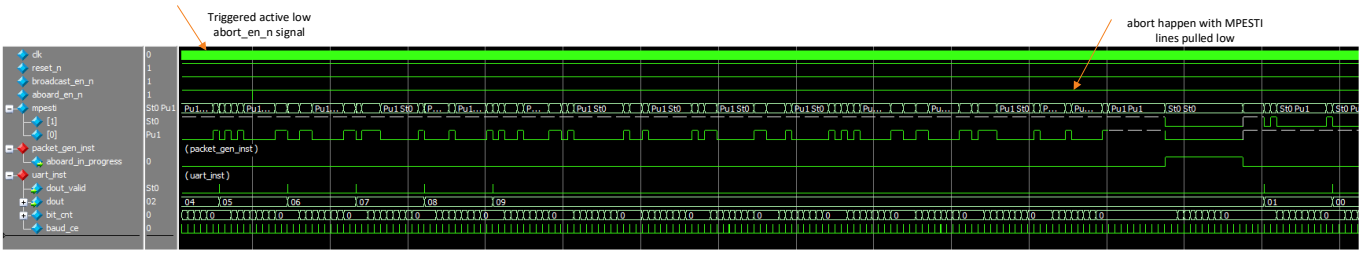


Figure 2.13. Abort Assertion

2.3. User Flow

The MPESTI Initiator comes with an APB collector (subordinate) user interface where it is used to initialize the MPESTI Initiator register settings, read the FIFO status, read the Discovery Payload data bytes and write/read the VW Payload data bytes. Refer to CSR Register Description (Initiator IP Core) section for more information. Figure 2.14 shows the MPESTI Initiator operation flowchart.

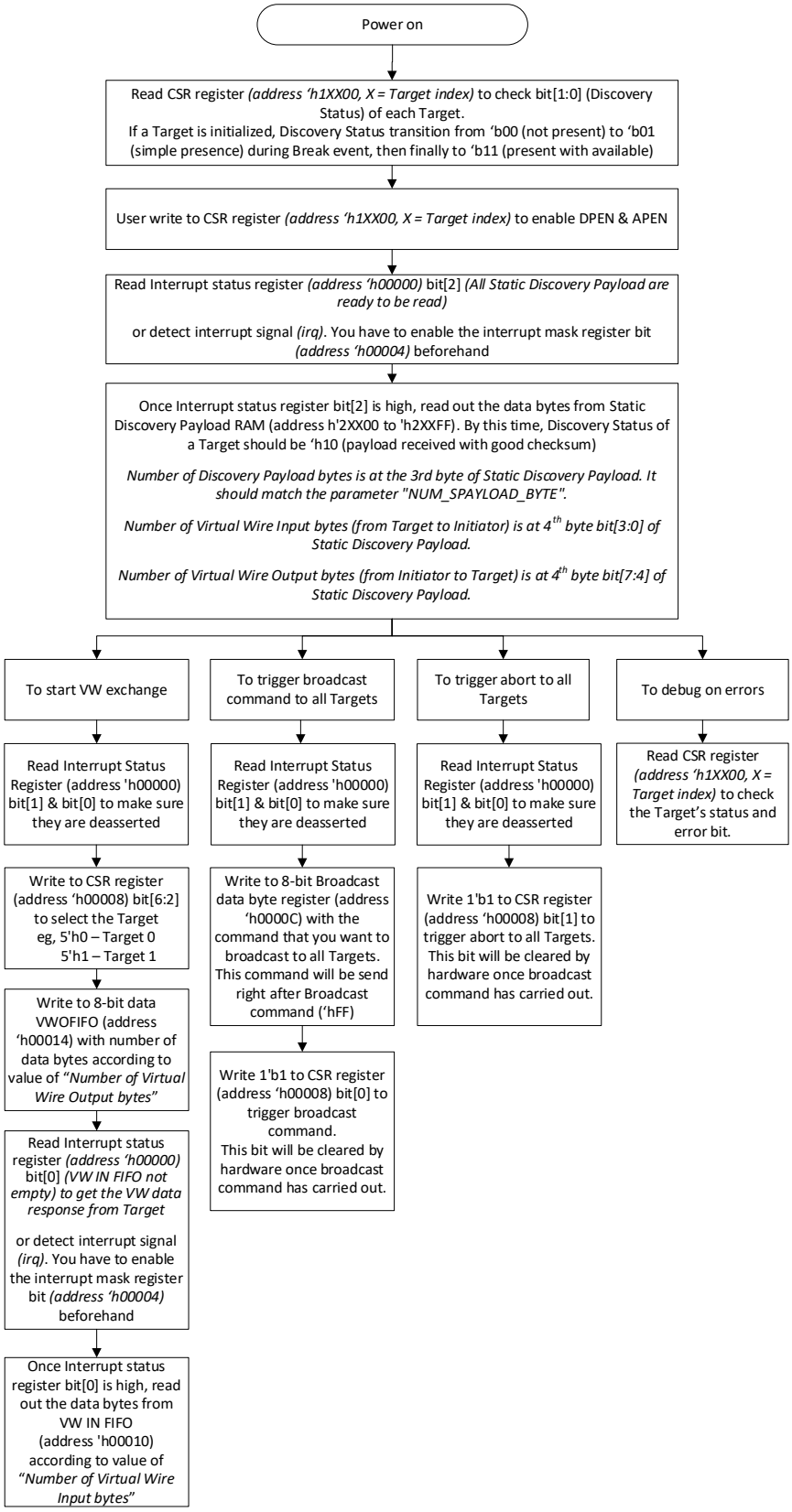


Figure 2.14. MPESTI Initiator IP User Flow

3. Reference Design Parameters

If the user wants to modify the default value, perform the following:

- Hardware – Change the parameter value at mpesti_top.sv
- Simulation – Change the parameter value at mpesti_top_tf.sv

Table 3.1. Reference Design Parameters

Name	Default	Range	Description
CLK_FREQ_HZ	12000000	—	Set this clock frequency to the value matching the input clock source from the development board.
NUM_TARGET	2	1–8	Number of MPESTI Target that the Initiator is connecting to. **Note:** If the user changes the default value, make sure to instantiate the mpesti_target_checker module and mpesti_target_top module in mpesti_top.sv per NUM_TARGET times. User must also create target_**_rom_mpesti_target_rom_copy.hex file under /source/ per NUM_TARGET times.
NUM_VW_IN_BYTE[TARGET-1:0]	{3, 8}	1–15	Number of VW Data byte response issued by MPESTI Targets. This parameter is an array per NUM_TARGET. Example: {<for target 1>, <for target 0>} **Note:** If the user wants to change the default value, user has to change the bit[3:0] value of fourth data byte in target_**_rom_mpesti_target_rom_copy.hex.
NUM_VW_OUT_BYTE[TARGET-1:0]	{12, 10}	1–15	Number of VW Data byte issued by MPESTI Initiator to MPESTI Targets. This parameter is an array per NUM_TARGET. Example: {<for target 1>, <for target 0>} **Note:** If user wants to change the default value, user has to change the bit[7:4] value of fourth data byte in target_**_rom_mpesti_target_rom_copy.hex.
NUM_SPAYLOAD_BYTE	16	16, 24, 32, 40, 48, 56, 64	Number of Static Discovery Payload Data byte response issued by MPESTI Targets. This parameter determines how many bytes are reserved for each MPESTI Target in the Static Discovery Payload RAM.

4. MPESTI Initiator IP Core Parameters

Table 4.1. MPESTI Initiator IP Core Parameters

Display Name	Default	Range	Description
Input Clock Frequency (HZ)	25000000	-	IP core clock frequency
Enable OSSP support	Disable	Disable, Enable	To enable or disable the Open Server Standard Program supported feature. When this feature is enabled, an additional input port called "flexio_check" is added. If an MPESTI line is in x4 bifurcation mode, the Flexio_check port is driven low. This feature checks each MPESTI line bifurcation mode and reflects the information to Discovery Payload byte 13.
Enable Virtual Wire support	Enable	Disable, Enable	To enable or disable the Dynamic Virtual Wire data exchange feature during the Active Phase. This feature is only usable if the Initiator is connected to an MPESTI Target (not simple presence mode).
Number of Static Discovery Payload Byte	16	16 – 64	The total number of Static Discovery Payload Data bytes returned by MPESTI Targets. This parameter specifies how many bytes are reserved in the Static Discovery Payload RAM for each MPESTI Target.
Number of Target lines	1	1 – 32	The total number of MPESTI line[s] to which the initiator is connected.
Memory File	-	For user to specify	Memory Initialization file for the Static Discovery Payload RAM. This file contains the default value of the Discovery Payload bytes for all MPESTI targets. The file format is in .hex. For this RD, the memory initiation file is pointing to "../../../source/initiator_mpesti_spayload_ram.hex"

5. System Pin Description

Table 5.1. Clock and Reset Interface

Name	Width	Direction	Description
clk	1	input	Clock signal
reset_n	1	input	Active low reset signal.

Table 5.2. MPESTI Interface

Name	Width	Direction	Description
mpesti_i	[NUM_TARGET-1:0]	Inout	This is a size configurable MPESTI line signal depending on parameter NUM_TARGET. These signals connect to MPESTI Initiator.
mpesti_t	[NUM_TARGET-1:0]	Inout	This is a size configurable MPESTI line signal. Each bit connects to the MPESTI target.

Table 5.3. UART Interface

Name	Width	Direction	Description
txd	1	output	This serial interface is driven by the Initiator Pattern Generator to display the Static Discovery payload and VW data byte on PC monitor.
rxd	1	input	This serial interface connects to the Initiator Pattern Generator. Currently the system is not expecting any input from the PC.

Table 5.4. Virtual Wire Interface

Name	Width	Direction	Description
gpi_8bit_data	8	input	These ports are used to allow user to send virtual wire data to the MPESTI Target 0.
vw_in_byte_0	8	output	These ports are used by the MPESTI Target 0 to receive virtual wire data from Initiator and display it on peripheral device.
vw_in_byte_1	8	output	These ports are used by the MPESTI Target 1 to receive virtual wire data from Initiator and display it on peripheral device.

Table 5.5. Conduit

Name	Width	Direction	Description
broadcast_en_n	1	input	Active low signal to send broadcast command request from MPESTI Initiator to MPESTI Targets.
abort_en_n	1	input	Active low signal to trigger abort event from MPESTI Initiator to MPESTI Targets.
led	1	output	This output signal is driven by a count up counter. It connects to the user LED (blinking LED) to indicate that the system clock is running correctly.

6. CSR Register Description (Initiator IP Core)

Access thru APB interface

Table 6.1. CSR Register Description (Initiator IP Core)

<table>
<tr><th>Byte Address</th><th>Name</th><th>Data width</th><th>Attribute</th><th>Description</th></tr>
<tr><td>‘h00000</td><td>FIFOs Status register</td><td>5</td><td>RO</td><td>Indicates the empty and full status of VWIFIFO, VWOFIFO and Static Discovery Payload FIFO

<table><tr><th>Bit</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr><tr><td>Description</td><td>VW OUT FIFO full</td><td>VW IN FIFO full</td><td>All Discovery SPAYLOAD are ready to be read</td><td>VW OUT FIFO NOT empty</td><td>VW IN FIFO NOT empty</td></tr></table></td></tr>
<tr><td>‘h00004</td><td>Interrupt Status Enable register</td><td>5</td><td>R/W</td><td>The IP core’s interrupt signal triggers based on FIFOs Status Register bits. Each bit in this register corresponds to a unique enable for the status register bits in register ‘h00</td></tr>
<tr><td>‘h00008</td><td>Control register</td><td>7</td><td>R/W</td><td>To trigger broadcast command or abort event on all MPESTI Targets.
To select MPESTI Target for VW abort exchange

<table><tr><th>Bit</th><th>[6:2]</th><th>1</th><th>0</th></tr><tr><td>Description</td><td>To select the MPESTI target before initiating the VW payload command.</td><td>Trigger abort mechanism</td><td>Trigger broadcast power break
(No response byte will be return by Target)</td></tr></table></td></tr>
<tr><td>‘h0000C</td><td>Broadcast data byte register</td><td>8</td><td>R/W</td><td>Users need to write to this register before enabling broadcast bit at register ‘h08. After transmitting the broadcast command, ‘hFF, Initiator will continue to send this data byte to all Targets.</td></tr>
<tr><td>‘h00010</td><td>VWIFIFO</td><td>8</td><td>RO</td><td>Virtual Wire Input FIFO. It is used to store the data bytes received from the M-PESTI Targets during active state. User can read out these data bytes through the APB interface.</td></tr>
<tr><td>‘h00014</td><td>VWOFIFO</td><td>8</td><td>WO</td><td>Virtual Wire Output FIFO. It is used to store the data bytes received from APB interface where these data bytes will be sent to the M-PESTI Targets during active state.</td></tr>
<tr><td>‘h1XX00
where, X is based on M-PESTI Target’s index number

Maximum number of Target = 32</td><td>MPESTI Target Error & Status register</td><td>8</td><td>R/W</td><td>This register is the error and status register of each MPESTI Target.

<table><tr><th>Bit</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>[1:0]</th></tr><tr><td>Description</td><td>CRC error</td><td>Parity error</td><td>APERR</td><td>DPERR</td><td>APEN</td><td>DPEN</td><td>DSTAT</td></tr></table>Eg, when APB address = 32’h00010000 -> MPESTI Target #0
APB address = 32’h00010100 -> MPESTI Target #1
APB address = 32’h00010200 -> MPESTI Target #2
….
APB address = 32’h00010F00 -> MPESTI Target #15</td></tr>
</table>

Byte Address	Name	Data width	Attribute	Description
'h2XX00 'h2XX04 'h2XX08 to 'h2XXFF (byte addressing) where, X is based on M-PESTI Target's index number Maximum number of Target = 32	Static Discovery Payload RAM	8	RO	Static Discovery Payload RAM. It is used to store the data bytes received from the M-PESTI Targets during Discovery State. During Discovery phase, M-PESTI initiator triggers Static Discovery Payload Request command to all the available MPESTI Targets in round robin manner. Then, M-PESTI Target responds by returning Static Discovery Payload bytes. Static payload bytes from all the Targets will be stored in the RAM in sequence. Eg, when APB address = 32'h00020000 -> MPESTI Target #0 to RAM byte address 0 APB address = 32'h00020104 -> MPESTI Target #1 to RAM byte address 4 APB address = 32'h00020204 -> MPESTI Target #2 to RAM byte address 4 .... APB address = 32'h00020F3C -> MPESTI Target #15 to RAM byte address 3C

7. Packaged Design

The reference design folder for MPESTI Initiator contains six subfolders:

- Document – contains the MPESTI Initiator Reference Design document.
- Project – contains the following subfolders:
  - Lattice Radiant™ – MachXO5-NX Radiant SW project files
  - Lattice Diamond® – MachXO3D Diamond SW project files
- Source – contains MPESTI reference design test components such as MPESTI Initiator Generator source files, MPESTI Target source files, RD top wrapper, MPESTI Initiator Static Discovery Payload RAM Initialization file (.hex).
- Testbench – contains simulation testbench file.
- Simulation – contains the modelsim.mdo file that launches the simulation run.
- Misc – contains the python script to be used by Raspberry Pi 4B minicomputer.
- IPK - contains the MPESTI Initiator IP ipk_gen.tcl. It can be used in Radiant and Propel to generate the IP core.

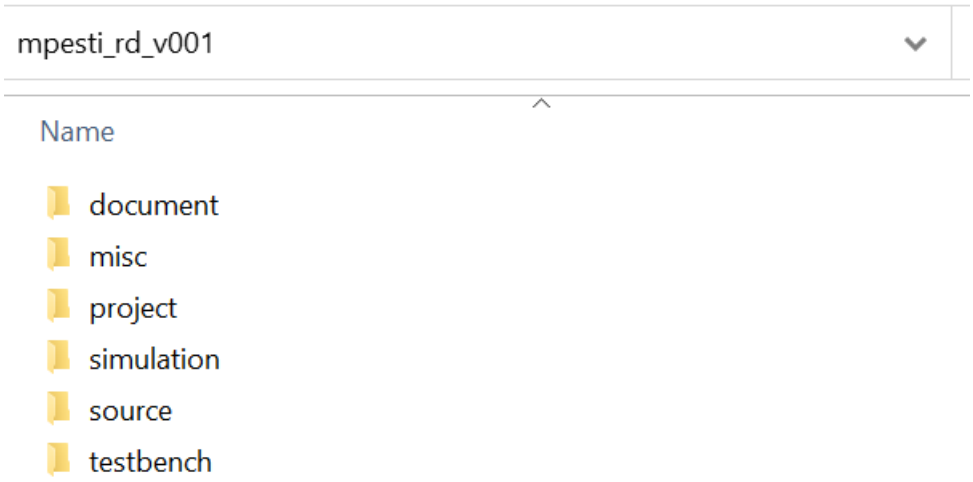


Figure 7.1. Packaged Design Directory Structure

8. Simulation and Verification

The reference design is simulated using the simulation environment in the testbench folder of the design package. The reference design contains one MPESTI pattern generator + checker, one MPESTI Initiator, and two MPESTI Targets + checkers.

To use the simulation file, perform the following steps:

1. Launch the ModelSim simulator.
2. Type *cd /simulation* in the ModelSim transcript window.
3. Type *do mpesti_sim.mdo* in the ModelSim transcript window.
4. Once the simulation has successfully completed, the window displays the *Simulation Passed* message. Otherwise, it displays the *Simulation Failed* message or *Error* message.

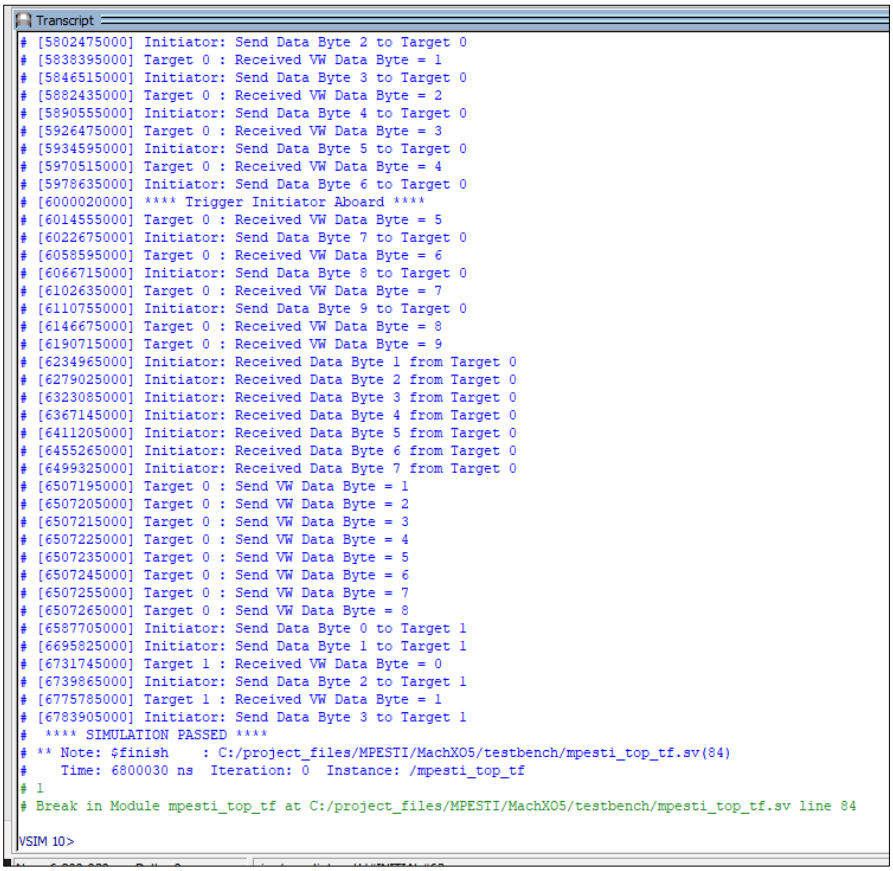


Figure 8.1. Simulation Printout

9. Resource Utilization

Table 9.1. Resource Utilization

MachXO5-NX Resource Usage	LUT4	Registers	Oscillator	EBR	I/O Buffers	GSR
	1991	1244	0	6	35	1

10. Migrating the Reference Design to Other Devices

To migrate the reference design to other devices, perform the following steps:

1. Open the existing project and change the device setting.

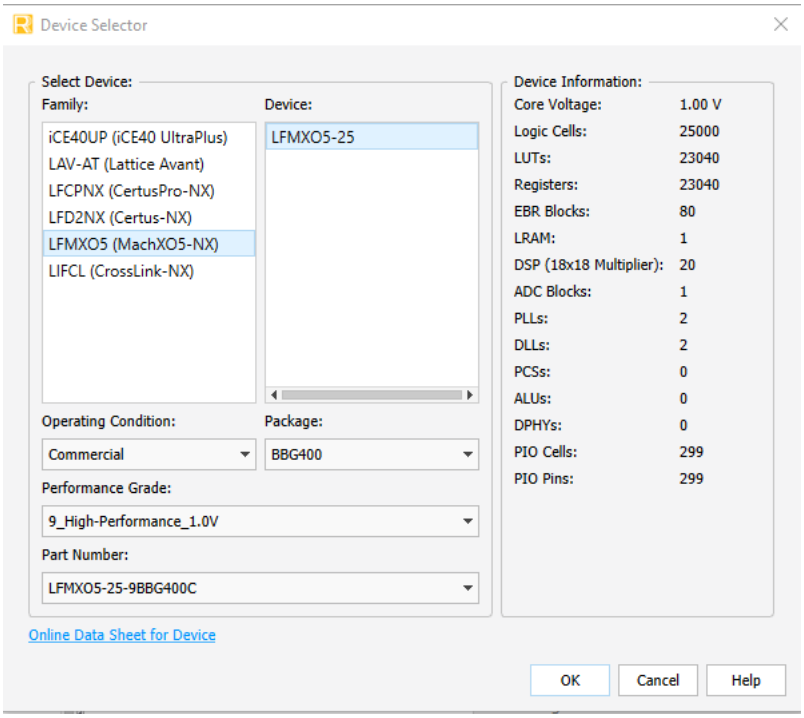


Figure 10.1. Device Selector Window

2. Run the **Synthesize Design** and provide the pin assignment in the **Device Constraint Editor**.

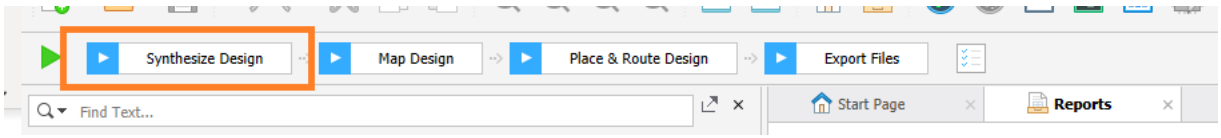
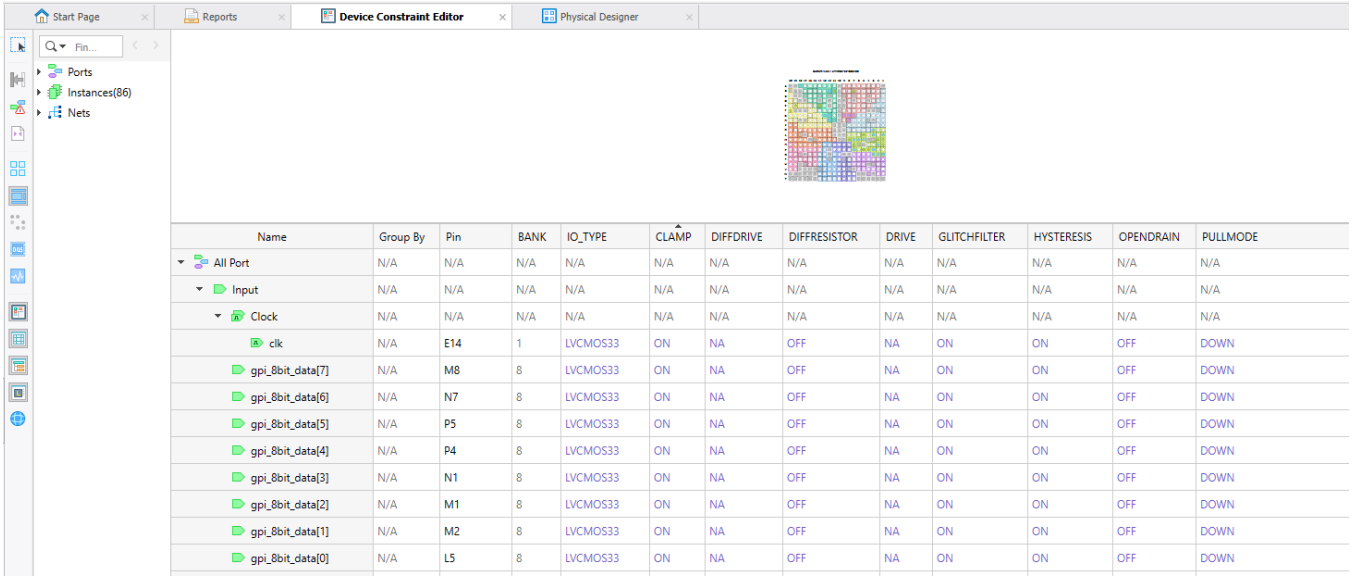


Figure 10.2. Synthesize Design



Name	Group By	Pin	BANK	IO_TYPE	CLAMP	DIFFDRIVE	DIFFRESISTOR	DRIVE	GLITCHFILTER	HYSTERESIS	OPENDRAIN	PULLMODE
All Port	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Input	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Clock	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
clk	N/A	E14	1	LVCMOS33	ON	NA	OFF	NA	ON	ON	OFF	DOWN
gpi_8bit_data[7]	N/A	M8	8	LVCMOS33	ON	NA	OFF	NA	ON	ON	OFF	DOWN
gpi_8bit_data[6]	N/A	N7	8	LVCMOS33	ON	NA	OFF	NA	ON	ON	OFF	DOWN
gpi_8bit_data[5]	N/A	P5	8	LVCMOS33	ON	NA	OFF	NA	ON	ON	OFF	DOWN
gpi_8bit_data[4]	N/A	P4	8	LVCMOS33	ON	NA	OFF	NA	ON	ON	OFF	DOWN
gpi_8bit_data[3]	N/A	N1	8	LVCMOS33	ON	NA	OFF	NA	ON	ON	OFF	DOWN
gpi_8bit_data[2]	N/A	M1	8	LVCMOS33	ON	NA	OFF	NA	ON	ON	OFF	DOWN
gpi_8bit_data[1]	N/A	M2	8	LVCMOS33	ON	NA	OFF	NA	ON	ON	OFF	DOWN
gpi_8bit_data[0]	N/A	L5	8	LVCMOS33	ON	NA	OFF	NA	ON	ON	OFF	DOWN

Figure 10.3. Device Constraint Editor

- For input ports:
  a. Assign clk to 12 MHz on board oscillator.
  b. Assign broadcast_en_n to SW2 User Push Button, set PULLMODE to UP.
  c. Assign abort_en_n to SW4 User Push Button, set PULLMODE to UP.
  d. Assign reset_n to SW3 User Push Button, set PULLMODE to UP.
  e. Assign gpi_8bit_data[7:0] to Raspberry Pi Connector.
    - gpi_8bit_data[0] – RASP_IO17
    - gpi_8bit_data[1] – RASP_IO10
    - gpi_8bit_data[2] – RASP_IO9
    - gpi_8bit_data[3] – RASP_IO11
    - gpi_8bit_data[4] – RASP_IO13
    - gpi_8bit_data[5] – RASP_IO19
    - gpi_8bit_data[6] – RASP_IO20
    - gpi_8bit_data[7] – RASP_IO21
  f. Assign rxd to RS232_RX_TTL (MachXO5) or any GPIO (MachXO3D). Set PULLMODE to UP.
- For output ports:
  a. Assign txd to RS232_TX_TTL (MachXO5), or any GPIO (MachXO3D).
  b. Assign led to User LED. This blinking LED is to indicate that the example design is programmed successfully to the FPGA.
  c. Assign vw_in_byte_0[7:0] to Raspberry Pi Connector.
    - vw_in_byte_0[0] – RASP_IO14
    - vw_in_byte_0[1] – RASP_IO15
    - vw_in_byte_0[2] – RASP_IO18
    - vw_in_byte_0[3] – RASP_IO23
    - vw_in_byte_0[4] – RASP_IO24
    - vw_in_byte_0[5] – RASP_IO25
    - vw_in_byte_0[6] – RASP_IO12
    - vw_in_byte_0[7] – RASP_IO16
  d. Assign vw_in_byte_1[7:0] to User LED
- For Bidir ports:
  a. Assign mpesti_t_0, mpesti_t_1, and mpesti_i[1:0] to PMOD, set PULLMODE to UP.

11. Setting up the Hardware

11.1. Setting up the MachXO5-NX LFMXO5-25 9BBG400C Development Board

To setup the board, perform the following steps:

1. Connect the jumper on JP10 to get the 12 MHz clock source from the FTDI chip.
2. Connect the 40-pin ribbon cable on the Raspberry Pi J6 header to the Raspberry Pi board GPIO header.
3. Loop back mpesti_t_0 pin to mpesti_i[0] pin with physical wire/jumper.
4. Loop back mpesti_t_1 pin to mpesti_i[1] pin with physical wire/jumper.
5. Connect the monitor and mouse to the Raspberry Pi USB ports.
6. Connect the mini-USB cable to J11.
7. After programming the .bit file, plug out the mini-USB cable from J11 and move the mini-USB cable to J19. The UART ports are connecting to the soft JTAG/UART user interface (port B) of the MachXO5NX Development Board.
8. Download the ComDebug application (Serial Communication Software for RS232, RS485 and Modbus Data Logging - Windmill ComDebug) to monitor the UART output display..

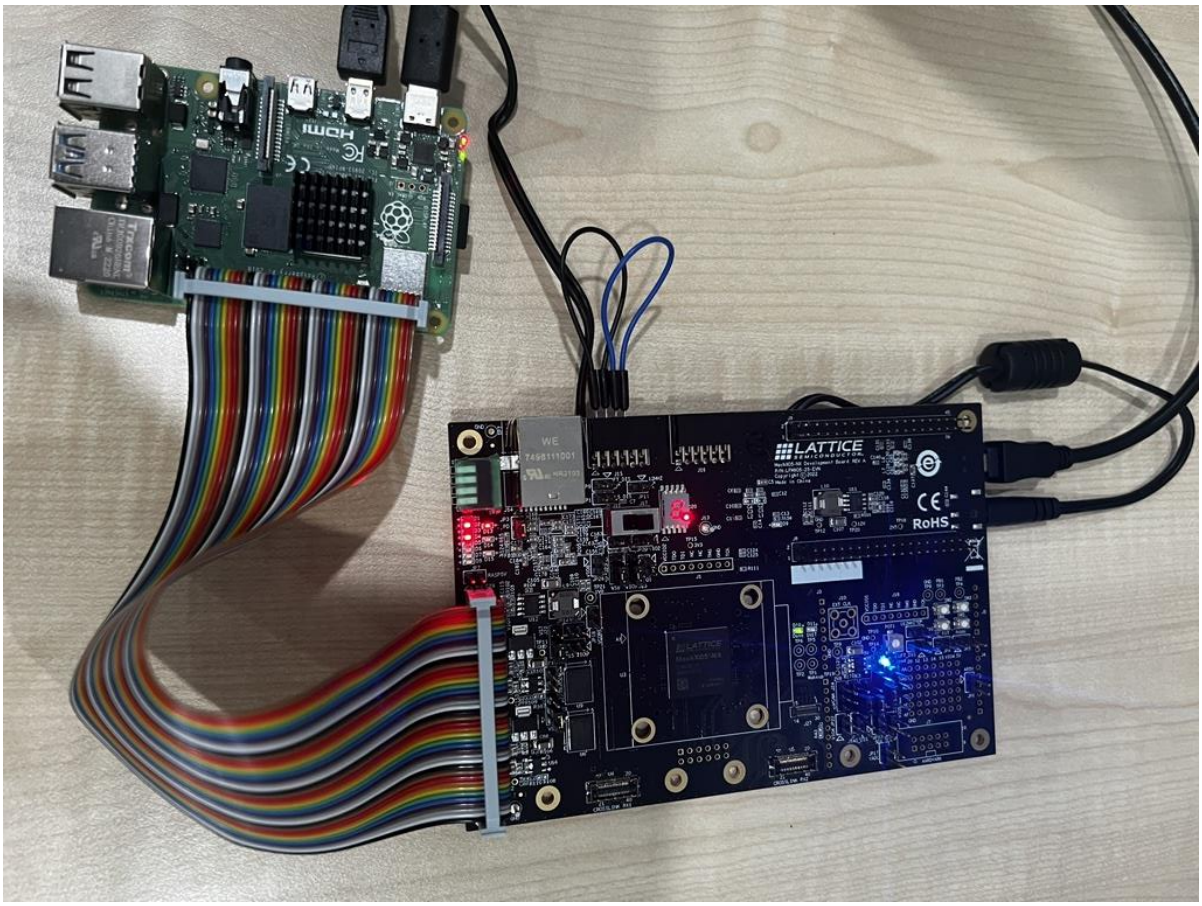


Figure 11.1. MachXO5-NX LFMXO5-25 9BBG400C Development Board Reference Design

11.2. Setting up the MachXO3D LCMXO3D-9400HC Development Board

To setup the board, perform the following steps:

1. Connect the jumper on JP11 for the 12 MHz clock source.
2. Connect the 40-pin Raspberry Pi J6 header on the Raspberry Pi board GPIO header.
3. Loop back mpesti_t_0 pin to mpesti_i[0] pin with physical wire/jumper.
4. Loop back mpesti_t_1 pin to mpesti_i[1] pin with physical wire/jumper.
5. Connect the mini-USB cable J11.
6. Connect RXD and TXD pin to a USB to UART converter and plug it to PC.
7. Download the ComDebug application (Serial Communication Software for RS232, RS485 and Modbus Data Logging - Windmill ComDebug) to monitor the UART output display..

References

For more information refer to:

- Avant-E web page
- MachXO5-NX web page
- MachXO3D web page
- MachXO3 web page
- Lattice Radiant FPGA design software
- Lattice Diamond FPGA design software
- Serial Communication Software for RS232, RS485 and Modbus Data Logging - Windmill ComDebug
- Lattice Insights 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, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 1.2, October 2023

Section	Change Summary
MPESTI Initiator IP core	Added this section.
All	Changed the following terms based on inclusive language. • Changed Master to Manager. • Changed Slave to Subordinate.

Revision 1.1, August 2023

Section	Change Summary
TOC	Changed the word *Aboard* to *Abort*.
Introduction	Updated 1.2 Limitations to The MPESTI Initiator IP core supports MachXO5-NX, MachXO3D, MachXO3L, MachXO3LF and Avant devices.

Functional Description	• Changed the word FIFO to RAM in subsection 2.1 Overview.
	• Rephrased the statement from This counter value is used as the Virtual Wire (VW) output data bytes that is written into the VW OUT FIFO in Initiator core, then sends out to MPESTI Target as virtual wire data byte, to This counter value is used to indicate the Virtual Wire (VW) output data bytes that is written into VW OUT FIFO register in the Initiator core, in subsection 2.1 Overview.
	• Changed VW IN FIFO to VW IN FIFO register in subsection 2.1 Overview.
	• Moved the step by step procedure of the ComDebug serial terminal application before the end of subsection 2.1 Overview.
	• Changed the word *Aboard* to *Abort* in 2.2.4 Abort transaction.
	• Updated Figure 2.14. MPESTI Initiator IP User Flow with following:
	  • Changed the word Aboard to Abort.
	  • Removed the statement to get the VW data response from Target after the word Read Interrupt status register (address ‘h00000) bit[0] (VW IN FIFO not empty).
	  • Updated CSR register address from ‘h2X to ‘h1XX00.
	  • Updated CSR register address from 'h08 bit[5:2] to 'h00008 bit[6:2].
	  • Updated CSR register address 'h08 to ‘h00008.
	  • Updated Interrupt status register address from ‘h00 to ‘h00000.
	  • Updated the Interrupt mask register bit address from ‘h01 to ‘h00004.
	  • Updated eg 4'h0 to 5'h0.
	  • Updated eg 4'h1 to 5'h1.
	  • Updated VWOFIFO address from ‘h14 to ‘h00014.
	  • Updated the VW IN FIFO address from 'h10 to 'h00010.
	  • Updated the 8-bit broadcast data byte register address from 'h0C to ‘h0000C.
	  • Replaced Static Discovery Payload FIFO not empty to All Static Discovery Payload are ready to be read.
	  • Replaced FIFO status register to Interrupt Status Register.
	  • Replaced Statis Discovery Payload FIFO (address 'h18') to Static Discovery Payload RAM (address ‘h2XX00 to 'h2XXFF).
	  • Replaced the statement User has to extract the Static Discovery Payload Size at 3rd byte of Static Discovery Payload to determine the start and end of the Payload data bytes for a particular Target to Number of Discovery Payload bytes is at the 3rd byte of Static Discovery Payload. It should match the parameter NUM_SPAYLOAD_BYTE.
	  • Replaced FIFO status register (address ‘h00) to Interrupt Status Register (address 'h00000).

Section	Change Summary

Functional Description	• Added the new flow Read Interrupt Status Register (address 'h00000) bit[1] & bit[0] to make sure they are deasserted below the following boxes:
	  • To trigger broadcast command to all Targets
	  • To trigger abort to all Targets
	  • To start VW exchange
	• Rephrased the statement from the value of this byte represents the SIZE/8, to the value of STATIC_PAYLOAD_SIZE represents the number of [Discovery Payload data bytes]/8.
	• Changed 16bytes to 16 discovery payload data bytes.

Reference Design Parameter	• Updated the Num_Target range from 1-8 to 1-32. • Added a new row for NUM_SPAYLOAD_BYTE reference design parameters. • Changed aboard_en_n to abort_en_n. • Changed the word Aboard to Abort.
CSR Register Description (Initiator IP Core)	• Updated values of the entire Table 6.1. CSR Register Description (Initiator IP Core).. • Rephrased the statement, User enables each of this register bit to turn on the interrupt triggering on register ‘h00, to Each bit in this register corresponds to a unique enable for the status register bits in register 'h00.
Packaged Design	• Updated the meaning of Source from contains MPESTI Initiator IP source code and reference design test components to contains MPESTI reference design test components such as MPESTI Initiator Generator source files, MPESTI Target source files, RD top wrapper, MPESTI Initiator Static Discovery Payload RAM Initialization file (.hex). • Added IPK definition
Resource Utilization	• Updated LUT4 value from *1623* to *1991*. • Updated Registers value from *1329* to *1244.*
Migrating the Reference	Changed aboard_en_n to abort_en_n.
All	Minor editorial changes

Revision 1.0, June 2023

Section	Change Summary
All	Initial release.

www.latticesemi.com