M-PESTI Initiator IP

IP Version: v1.3.0

User Guide

FPGA-IPUG-02258-1.3

July 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.

Abbreviation	Definition
AHB	Advanced High-performance Bus
APB	Advanced Peripheral Bus
CPU	Central Processing Unit
CMD	Command
CRC	Cyclic Redundancy Check
FPGA	Field Programmable Gate Array
GPIO	General Purpose I/O
GUI	Graphical User Interface
IP	Intellectual Property
LUT	Look Up Table
M-PESTI	Modular-Peripheral Sideband Tunneling Interface
PDC	Physical Design Constraint
PLL	Phase-Locked Loop
RAM	Random-Access Memory
SoC	System-on-a-Chip
SRAM	Static Random-Access Memory
UART	Universal Asynchronous Receiver Transmitter

1. Introduction

1.1. Overview of the IP

The Modular-Peripheral Sideband Tunneling Interface (M-PESTI) Initiator IP core provides early peripheral presence detection and attribute collection before system boot up. This IP core supports bidirectional communication with initiator command and target response structure.

It is designed to comply with the Modular-Peripheral Sideband Tunneling Interface (M-PESTI) Base Specification Version 1.0 Release Candidate 2.

1.2. Quick Facts

Table 1.1. Summary of the M-PESTI Initiator IP

IP Requirements	Supported Devices	MachXO3™, MachXO3D™, Mach™-NX, MachXO5™-NX, Lattice Avant™, and Certus™-N2
	IP Changes	For a list of changes to the IP, refer to the M-PESTI Initiator IP Release Notes (FPGA-RN-02005).
Resource Utilization	Supported User Interface	Advanced Peripheral Bus (APB)
	Resources	See Appendix A. Resource Utilization
Design Tool Support	Lattice Implementation	IP core v1.3.0 - Lattice Radiant software 2025.1
	Synthesis	Lattice Synthesis Engine (LSE) Synopsys® Synplify Pro® for Lattice
	Simulation	For the list of supported simulators, see the Lattice Radiant Software User Guide.

1.3. IP Support Summary

Table 1.2. IP Support Summary of the M-PESTI Initiator IP

Device Family	System Clock	Number of Target Devices	Radiant Timing Model	Hardware Validated
MachXO5-NX	25 MHz	2	Final	Yes
	100 MHz	2	Final	No
MachXO3D	25 MHz	2	Final	No
	100 MHz	2	Final	No
MachXO3LF	25 MHz	2	Preliminary	No
	100 MHz	2	Preliminary	No
MachXO3L	25 MHz	2	Preliminary	No
	100 MHz	2	Preliminary	No
Certus-N2	25 MHz	2	Preliminary	No
	100 MHz	2	Preliminary	No
Lattice Avant	25 MHz	2	Preliminary	No
	100 MHz	2	Preliminary	No

1.4. Features

The following lists the key features of the M-PESTI Initiator IP core:

- Communicates via half-duplex bidirectional UART protocol at 250k baud rate, 8-bit data, 1-bit odd parity, 1 start bit and 1 stop bit M-PESTI port.
- Supports Static Discovery payload with CRC-8 payload checksum.
- Supports one initiator to many targets system.
- Supports configurable number of M-PESTI devices up to 64 targets.
- Supports autonomous Static Discovery Payload request command to all Targets on round robin manner during Discovery phase.
- Supports Static Discovery Payload request command retry. If the payload is not successfully received after an initial attempt, two more retries per target are triggered before proceeding to the next target(s). When the turn returns to the target, the initiator persistently send command attempts as a set of initial and two retry attempts until successful payload is received.
- Supports Target reset at any time.
- Supports sending broadcast command.
- Supports an aborting of ongoing discovery or active phase command to insert a broadcast command.
- Supports source and destination cable coupling discovery.

1.5. Licensing and Ordering Information

An IP-specific license string is required to enable full use of the M-PESTI Initiator IP core in a complete, top-level design.

The IP can be fully evaluated through functional simulation and implementation (synthesis, map, place and route) without an IP license string. This IP supports Lattice's IP hardware evaluation capabilities. You can create versions of the IP to operate in hardware for a limited time (approximately four hours) without requiring an IP license string. A license string is required to enable timing simulation and to generate a bitstream file that does not include the hardware evaluation timeout limitation.

For more information about pricing and availability of the M-PESTI Initiator IP core, contact your local Lattice Sales Office.

1.5.1. Ordering Part Number

Table 1.3. Ordering Part Number

Device Family	Part Number	
	Single Seat Annual	Single Seat Perpetual
MachXO5-NX	MPESTI-I-XO5-US	MPESTI-I-XO5-UT
MachXO3D	MPESTI-I-XO3D-U	MPESTI-I-XO3D-UT
MachXO3	MPESTI-I-XO3-US	MPESTI-I-XO3-UT
Mach-NX	MPESTI-I-MNX-US	MPESTI-I-MNX-UT
Certus-N2	MPESTI-I-CN2-US	MPESTI-I-CN2-UT
Avant-AT-G	MPESTI-I-AVG-US	MPESTI-I-AVG-UT
Avant-AT-X	MPESTI-I-AVX-US	MPESTI-I-AVX-UT
Avant-AT-E	MPESTI-I-AVE-US	MPESTI-I-AVE-UT

1.6. Hardware Support

Refer to the Example Design section for more information on the boards used.

1.7. Minimum Device Requirements

There is no limitation in device speed grade for M-PESTI Initiator IP core. For the minimum required resources to instantiate this IP and maximum clock frequency supported, see Appendix A. Resource Utilization section.

1.8. Naming Conventions

1.8.1. Nomenclature

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

1.8.2. Signal Names

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

2. Functional Description

2.1. IP Architecture Overview

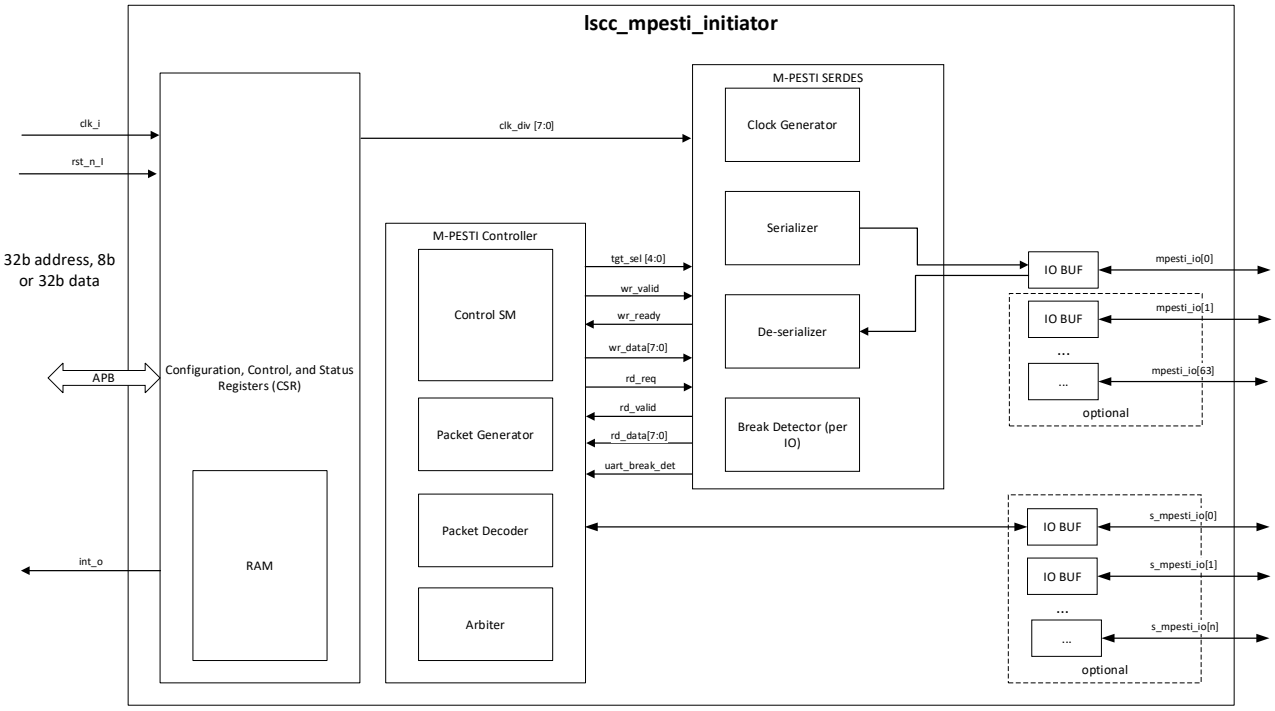

Figure 2.1. Lattice M-PESTI Initiator IP Core Block Diagram

The M-PESTI Initiator IP core includes the following layers:

- APB Interface
- Configuration, Control and Status Registers
- This block includes the registers for user command and configurations and target/s status and the internal RAM for the payload allocation.
- M-PESTI Initiator Top
- This block includes logics related to input and output ports and blocks instantiation.
- M-PESTI Controller
- This block includes the implementation of the process flow as well as the device status tracking and the generation and decoding of the necessary packet format. This also includes the implementation of the round robin servicing for multiple targets sharing one initiator.
- M-PESTI SERDES
- This block includes the implementation of the low level half-duplex UART protocol and the generation of the divided clock and cyclic redundancy check (CRC).

2.2. Clocking

There is one clock source for the M-PESTI Initiator IP core.

clk_i: System clock. Set the value in the *System Clock Frequency* attribute. This clock is used to generate the M-PESTI baud rate of 250 kHz for the M-PESTI wire transactions.

2.3. Reset

There is one hardware reset for the M-PESTI Initiator IP core.

rst_n_i: An asynchronous active low reset.

Note: This asynchronous reset is synchronized with the system clock. Upon the de-assertion of rst_n_i, three (3) clock cycles are needed to propagate the reset in the IP core blocks.

2.4. User Interfaces

The following table shows the user interfaces and supported protocols. The memory-mapped interface of M-PESTI Initiator IP core is APB interface only.

Table 2.1. User Interfaces and Supported Protocols

User Interface	Supported Protocols	Description
Memory-Mapped Interface	APB	Data width is set to 32 bits. The following describes the read and write transactions of the APB interface: • Write transaction has no wait states • Read transaction has one wait states For more information the APB interface and the timing diagrams, refer to the AMBA 3 APB Protocol v1.0 Specification.

2.5. M-PESTI Protocol

2.5.1. M-PESTI UART Frame

The M-PESTI wire is a simple half-duplex UART. Start bit is determined when the wire is pulled from high to low since the frame is asynchronous to the receiving clock. Following the start bit is the 8 bits data, odd parity, and one stop bit running at the 250 kHz +/- 3% baud rate. Stop bit is determined when the wire is pulled from low to high. When stop bit is zero (STOP_BIT=0), framing error will be flagged.

The next frame may start after stop bit of the previous frame is sampled at the positive edge of the receiving clock.

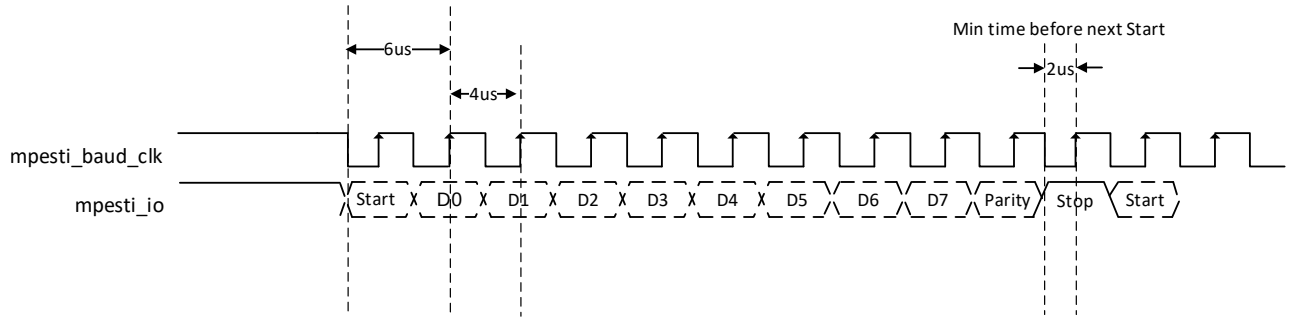

Figure 2.2. M-PESTI UART Frame

2.5.2. M-PESTI Protocol Phase

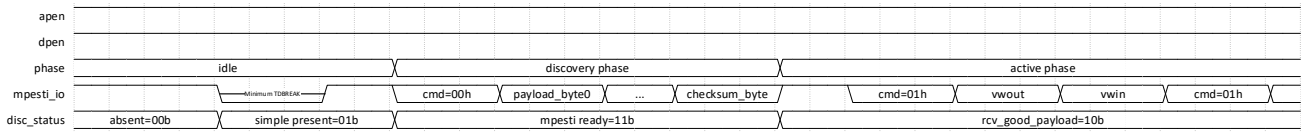

Figure 2.3. M-PESTI Protocol Timing Diagram

2.5.2.1. Target Presence Detection

Before the frame-based discovery, the initiator must detect the target presence. The event of target holding the M-PESTI wire low for greater than the frame length is called UART BREAK. The target must meet the minimum target discovery break width of 50us to guarantee presence detection. The maximum target discovery break width is configured through the *Target Discovery Break Assertion Time* attribute. Exceeding the maximum limit will trigger a target discovery break timeout.

2.5.2.2. Target Setting Reconfiguration

The initiator samples the configurations of the selected target during the transition of target select. Reconfiguration in between transaction takes effect after the current transaction.

2.5.2.3. Discovery Phase

After the target releases the UART BREAK, the initiator autonomously sends the discovery phase command, 8'h00, to an M-PESTI ready target given that the *Discovery Phase Enable* attribute is checked. On the other hand, when the *Discovery Phase Enable* attribute is unchecked, you can program the *Discovery Phase Enable* register to manually perform the discovery phase. When the target did not send a respond, the initiator resends the discovery phase command until good payload is successfully received. Re-discovery after a good payload is received can be done by writing 0 then 1 to the *Discovery Phase Enable* register or by hardware reset.

Target payload size is determined by its payload information, STATIC_PAYLOAD_SIZE (Payload Offset+0x02). This is limited from 8 bytes up to 64 bytes of data.

A single M-PESTI initiator can handle up to 64 targets. In the case of instantiating a single initiator to multiple targets, round robin servicing is implemented.

For discovery phase, if good payload is unsuccessfully received from the current target, it will continuously send the discovery phase command until the good payload is received successfully. Good payload means that there is no CRC error, parity error, and framing error. There is a limitation of three unsuccessful retries (initial attempt + two retries) before going to the next target, which asserts the Discovery Retry Rollover bit of Target Control Status.

The following figure shows the transition of the target arbitration given that there are three (3) M-PESTI ready targets instantiated, both discovery phase and active phase is configured as enabled and all targets release break at the same time. At the first arbitration cycle, initiator transacts to each target for the discovery phase. In this example, target select is at Target 0 and this target is M-PESTI ready. Target 0 reached 3 discovery retries. The initiator proceeds to the next target, Target 1. While Target 1 and Target 2 is doing the discovery phase, Target 0 stays at discovery phase waiting for payload request command. Then, Target 1 and Target 2 transitioned to an active phase after receiving a good payload and then waited for its turn to start virtual wire exchange.

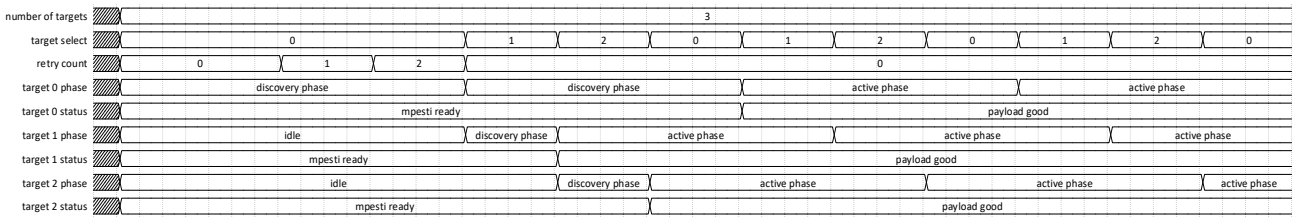

Figure 2.4. Single Initiator Multiple Targets Transaction

2.5.2.4. Active Phase

When APEN is enabled in the *Active Phase Enable* attribute and the discovery phase is successful meaning the initiator receives good checksum, the initiator autonomously sends the active phase command, 8'h01, then exchanges data to each target through the M-PESTI interface. When the *Active Phase Enable* attribute is unchecked, you can program the *Active Phase Enable* register to manually perform the virtual wire exchange. The initiator expects a respond within 500us. An active phase receive timeout error is flagged when there is no response from the target. The M-PESTI Initiator IP continuously send active phase command through round robin servicing to all targets unless Active Phase is disabled.

Current autonomous virtual wire exchange transacts virtual wire input and output data based on the *Number of Virtual Wire Input and Output Bytes* attribute per target.

When *Enable Programmable Virtual Wire* attribute is checked, this allows reconfiguration of the number of virtual wire bytes per target. However, the allowable reconfigurable values must not be greater than the values set on the *Number of Virtual Wire Input Bytes* and *Number of Virtual Wire Output Bytes* attributes.

If the number of virtual wire output bytes is zero, the initiator sends 8'h00 data. If the number of virtual wire input bytes is zero, the target is expected to send 8'h00 as an acknowledgement response.

To support manual multiple virtual wire bytes transactions, refer to the following figure that shows the manual program flow through user command.

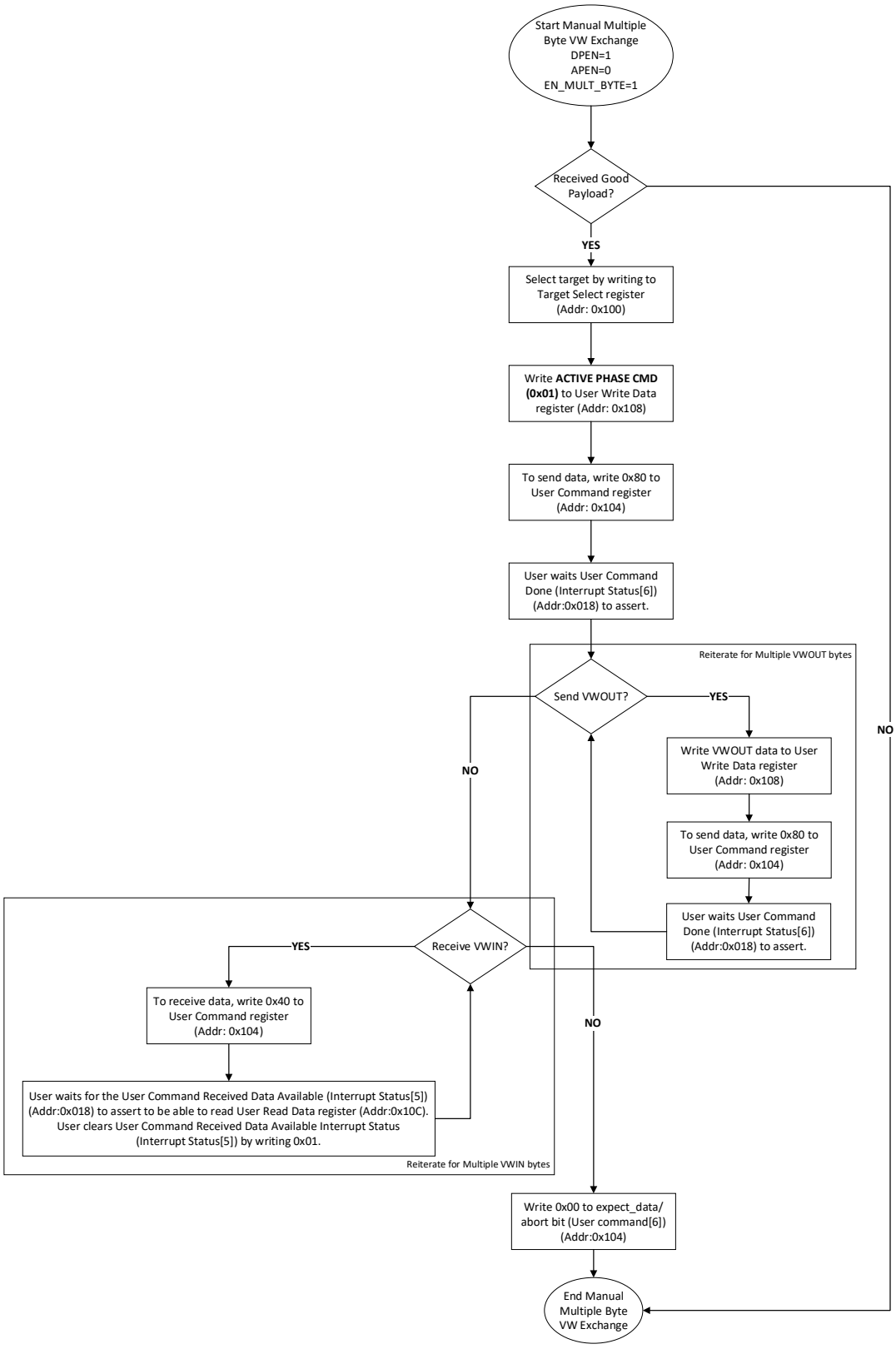

Figure 2.5. Manual Multiple Virtual Bytes Program Flow

2.5.2.5. User Command, Broadcast, and Abort Mechanism

For independent commands, User Command register can be configured to transact with the target. Enabling send data in the User Command register sends the data in the User Write data register. Additionally, enabling the expected data in the User Command register expects response from the target, which can be read in the User Read data register. The initiator expects the target responds within 250 ms. Otherwise, a receive timeout is asserted in the Interrupt Status register.

Because multiple targets can share a single initiator, this limits transactions at the same time. The broadcast command enables the initiator to send commands to multiple targets simultaneously. The initiator sends a broadcast command, 8'hFF, and the data to be broadcasted. This command has no target response.

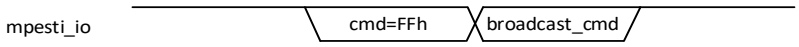

Figure 2.6. Broadcast Command

The M-PESTI initiator has an option to abort the current transaction or to continue before sending the broadcast command. Aborted transactions may happen during the initiator transmission and target transmission.

Figure 2.7. Abort Mechanism

User command can be programmed as shown in the following figure.

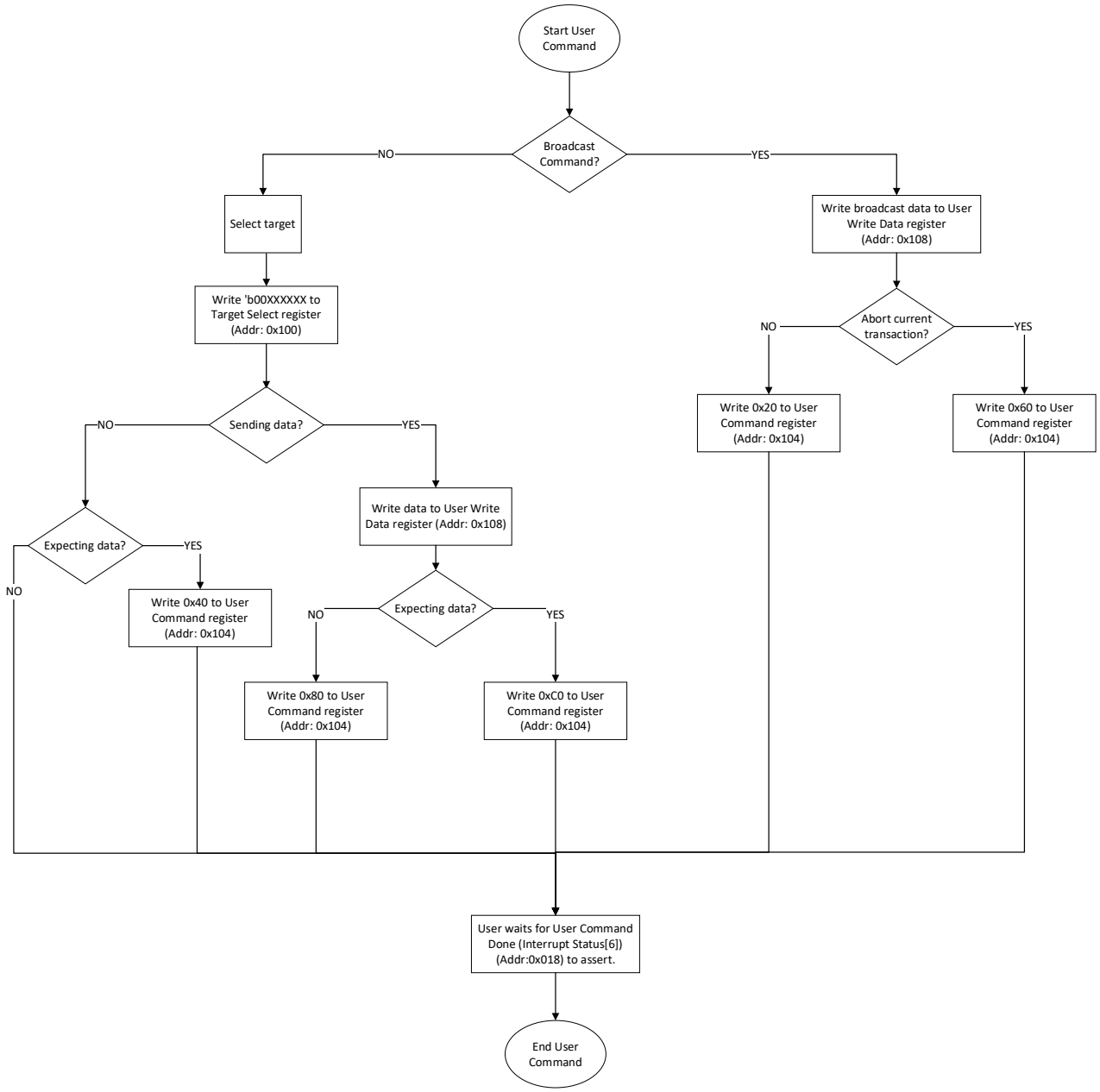

Figure 2.8. User Command Program Flow

2.5.2.6. Target Reset

If the target resets prior to a successful discovery phase, the initiator will autonomously send a discovery phase command to retry discovery given that the discovery phase is enabled.

If the target resets after a successful discovery phase, the discovery status and payload information is locked. The initiator will send an active phase command to initiate a virtual wire exchange. Because of the target resets, the active phase command is not observed and flags an active phase error due to timeout.

2.5.2.7. Secondary Wire Capability with Stimulus and Response

The primary source wire is used for frame-based communication. The initial discovery payload determines if there are additional source to destination coupling needed to be identified. These couplings are defined to be the secondary M-PESTI wires.

You reiterate the process of selecting wires and sending low (secondary wire stimulus is zero) stimulus to each secondary wire to detect all wires associated with that target.

The selectable value using the wire select is based on the configuration in the Total Number of Secondary Wire attribute. Selecting values greater than the configuration set in the Total Number of Secondary Wire attribute will be ignored. The commands are sent to the previously selected wire.

The following figure shows the Source Discovery Detection Flow of the secondary wires.

Note: N is the target number.

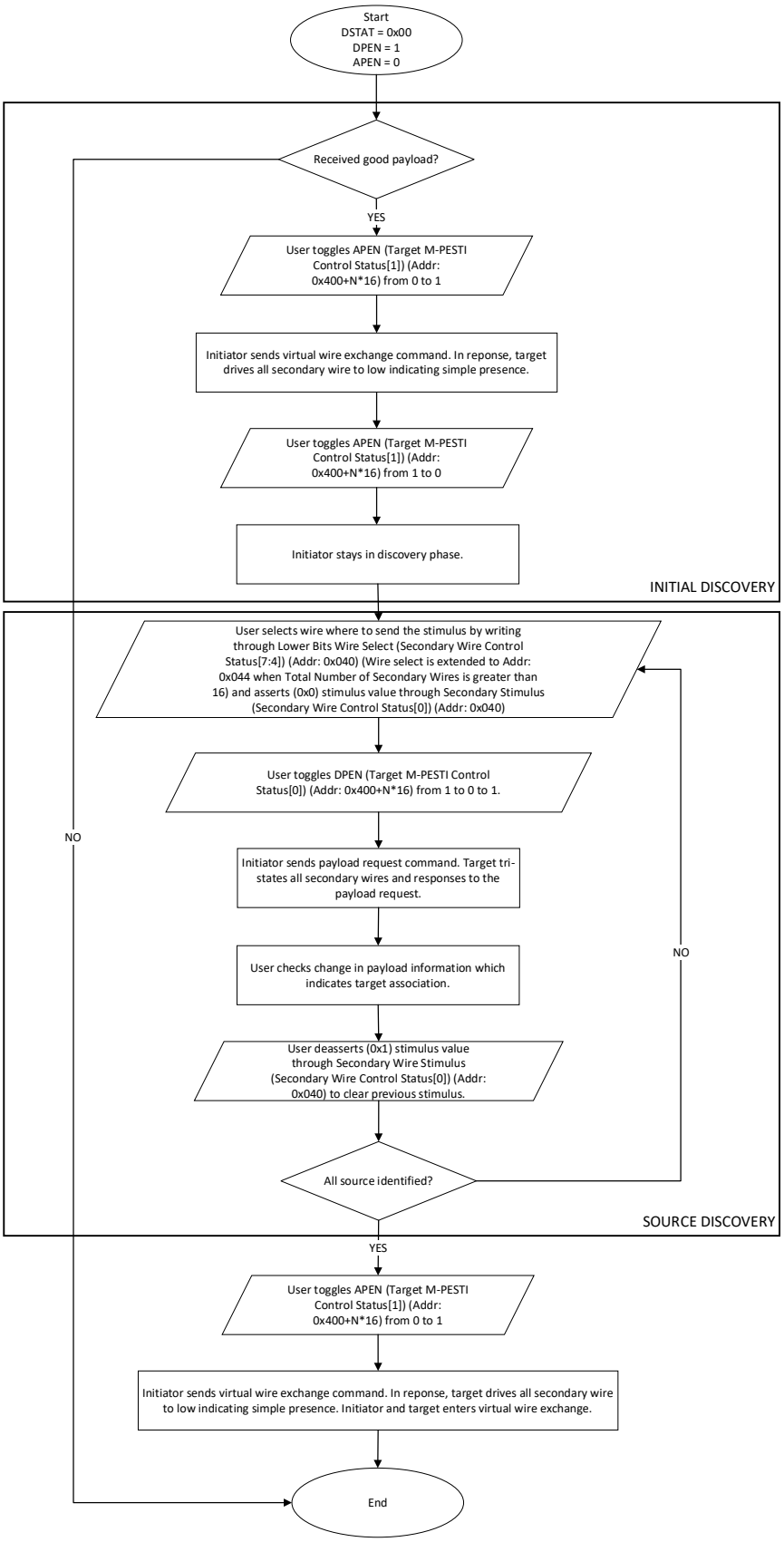

Figure 2.9. Secondary Wire Capability Program Flow

3. IP Parameter Description

The configurable attributes of the M-PESTI Initiator IP core are shown in the following tables. You can configure the IP by setting the attributes accordingly in the IP Catalog's Module/IP wizard of the Lattice Radiant software.

Wherever applicable, default values are in bold.

3.1. General

Table 3.1. General Attributes

Attribute	Selectable Values	Description
Clock Setting		
System Clock Frequency (MHz)	1-100, **25**	Specifies the input system clock of the IP.
System Clock Period (ns)	—	—
M-PESTI Baud Rate (kHz)	—	—
M-PESTI Baud Period (ns)	—	—
Calculated Clock Divider Setting	—	—
APB Interface Setting		
APB Data Width	—	Specifies the data width of the APB interface. Display only.
Target Setting		
TDBreak Assertion Time (us) (multiples of 2us)	**1,000**-2,000	Specifies the maximum target discovery break assertion time.
Number of Target Devices	**1**-64	Specifies the number of targets sharing one initiator.
Total Number of Secondary Wires	**0**-(24*Number of Target Devices)	Specifies the total number of secondary wires of all targets. The maximum number of secondary wires per target is 24. Example: Two targets are instantiated. Target 0 has two secondary wires and Target 1 has one secondary wire. Total Number of Secondary Wires attribute must be configured to 3. Note that each device has limited number of input or output ports.
Maximum Static Discovery Payload Size	8, 16, 32, **64**	Specifies the maximum size of the discovery payload among all targets.
Discovery Phase Enable	**Checked**, Unchecked	When checked, the discovery phase is enabled. This can be re-configured by writing to the Discovery Phase Enable register.
Active Phase Enable	**Checked**, Unchecked	When checked, the active phase is enabled. This can be re-configured by writing to the Active Phase Enable register.
Enable CSR for Secondary Wire	Checked, **Unchecked**	When checked, this generates a register— Secondary Wire Control Status [2] register—that contains secondary wire value and can be read.
Enable Glitch Filter	Checked, **Unchecked**	When checked, pulses below 50 ns are suppressed.
Allow Multiple User Receive Bytes	**Checked**, Unchecked	If checked and the expect_data/abort (User Command[6]) is enabled, this parameter allows you to receive multiple bytes of data. Otherwise, user command receives 1 byte.
Maximum User Receive Byte Size	**2**-16	Specifies the maximum byte size of user data that can be stored. It is recommended to use the default value to minimize LUT size. However, it is necessary to read received data within 88us.
Memory Setting		
Payload Memory Allocation Option (for single target only)	**RAM**, Register	Specifies where to store the payload if a single target is selected. Otherwise, the RAM is used for the payload allocation.
Payload Memory	—	—

Attribute	Selectable Values	Description
Allocation		
I/O Setting		
Include I/O Primitive	**Checked**, Unchecked	When checked, this includes the I/O Primitive instance. The M-PESTI ports, both primary and secondary, are seen as bidirectional I/O. Otherwise, they are seen as tristate M-PESTI ports.
Virtual Wire Setting		
Enable Programmable Virtual Wire	**Checked**, Unchecked	Allows reconfiguration of Number of Virtual Wire Input and Output Bytes per target. Enabling this attribute generates M-PESTI Target Virtual Wire Configuration registers.
Enable Virtual Wire as Ports (in hex)	**0** – Number of Target Devices	This is a bitwise configuration for enabling virtual wire as physical input and output ports. Input value must not be greater than the *Number of Target Devices* attribute.
Target x[1]		
Enable Virtual Wire as Ports x[1]	—	Reflects Enable Virtual Wire as Port for Target x. Display only.
Number of Virtual Wire Input Bytes	0-16, **1**	Specifies how many virtual wire input bytes for Target x.
Number of Virtual Wire Output Bytes	0-16, **1**	Specifies how many virtual wire output bytes for Target x.

Note:

1. Attributes below Target x group are configurable per target device where x is the device number which is limited by the Number of Target Devices attribute.

4. Signal Description

The following table lists the input and output signals for the M-PESTI Initiator IP core along with their descriptions.

Table 4.1. Ports Description

Port	Type	Description
System Clock and Reset		
clk_i	Input	This signal is the system clock in MHz defined in the *System Clock Frequency* attribute.
rst_n_i	Input	This signal is an asynchronous active low reset.
APB Interface		
apb_psel_i	Input	This is the APB interface select signal.
apb_paddr_i	Input	This is the APB interface address signal.
apb_pwdata_i	Input	This is the APB interface write data signal.
apb_pwrite_i	Input	This is the APB interface direction signal. • 1'b0: Read • 1'b1: Write
apb_penable_i	Input	This is the APB interface enable signal.
apb_pslverr_o	Output	This is the APB error signal. This signal is tied to 1'b0.
apb_pready_o	Output	This is the APB interface ready signal. Indicates transfer completion. The completer uses this signal to extend an APB transfer.
apb_prdata_o	Output	This is the APB interface read data signal.
M-PESTI Interface		
mpesti_io[1]	Inout	This signal is a bidirectional M-PESTI signal.
mpesti_i[1]	Input	This signal is an M-PESTI input signal.
mpesti_oe[1]	Input	This signal is an M-PESTI output enable signal.
mpesti_o[1]	Output	This signal is an M-PESTI output signal.
s_mpesti_io[1]	Inout	Generated when the *Total Number of Secondary Wires* attribute is not zero. This signal is a bidirectional secondary M-PESTI signal.
s_mpesti_i[1]	Input	Generated when the *Total Number of Secondary Wires* attribute is not zero. This signal is a secondary M-PESTI input signal.
s_mpesti_oe[1]	Input	Generated when the *Total Number of Secondary Wires* attribute is not zero. This signal is a secondary M-PESTI output enable signal.
s_mpesti_o[1]	Output	Generated when the *Total Number of Secondary Wires* attribute is not zero. This signal is a secondary M-PESTI output signal.
Virtual Wire Interface[2]		
vwire_out_tgt0	Input	This signal is the Virtual Wire Output of Target 0. Applicable when *Enable Virtual Wire as Ports* == Checked for Target 0. Note that the bus width is based on the *Number of Virtual Wire Output Bytes* attribute of the specific target.
vwire_in_tgt0	Output	This signal is the Virtual Wire Input of Target 0. Applicable when *Enable Virtual Wire as Ports* == Checked for Target 0. Note that the bus width is based on the *Number of Virtual Wire Input Bytes* attribute of the specific target.
Interrupt Interface		
int_o	Output	This signal is the interrupt signal.

Notes:

1. The *Include I/O Primitive* attribute states the M-PESTI interface ports.
2. This interface is generated when *Enable Virtual Wire as Ports* == Checked. This attribute can be set per target.

5. Register Description

The register address map starts at 0x000 and increment by 4 per register. For single M-PESTI target, the *Memory Allocation Option* attribute can be set to either **Register** or **RAM,** which indicates the location to store the payload. On the other hand, RAM is used when using multiple targets.

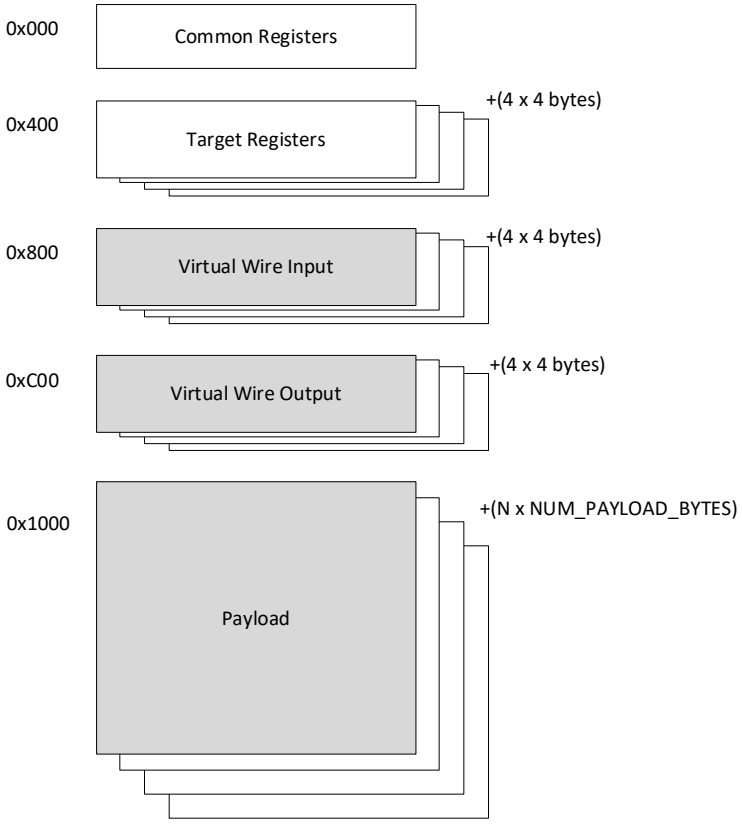

Figure 5.1. Memory Map Allocation

The M-PESTI status and control register addresses for multiple targets is computed as such where N is the target number:

0x400+N*(4 x 4 bytes) – **0x404**+N*(4 x 4 bytes)

Internal RAM depth depends on payload per target. The payload per target is configurable in multiples of 8 bytes and a maximum of 64 bytes). For example, four targets with 32 bytes payload each = 128 bytes RAM. The RAM offset starts at 0x800.

Address allocation for Virtual Wire Input and Virtual Wire Output registers with an offset of 0x800 and 0xC00 respectively.

0x800+N*(4 x 4 bytes) , **0xC00**+N*(4 x 4 bytes)

Example:

When two targets are instantiated, M-PESTI Target 0 register addresses are from 0x400 to 0x404 and M-PESTI Target 1 register addresses are from 0x410 to 0x414. Virtual Wire Input register addresses for Target 0 and Target 1 is 0x800 and 0x810, respectively. Then, for Virtual Wire Output register addresses, 0xC00 for Target 0 and 0xC10 for Target 1.

Table 5.1. Register Access Types

Access Type	Access Type Abbreviation	Behavior on Read Access	Behavior on Write Access
Read only	RO	Returns register value.	Ignore write access.
Write only	WO	Returns 0.	Updates register value.
Read and write	RW	Returns register value.	Updates register value.
Read and write 1 to clear	RW1C	Returns register value.	Writing 1'b1 on register bit clears the bit to 1'b0. Writing 1'b0 on register bit is ignored.

Note: All bits without field assignments are treated as Reserved with read-only access.

Table 5.2. Register Address Map

Offset APB	Register Name	Access Type	Description
Common Registers			
0x000	Reserved	RO	Reserved address.
0x004	Configuration	RW	Sets the configuration of the clock divider.
0x008	Software Reset	RW	Sets the software reset of the IP core and the CSR.
0x00C-0x014	Reserved	RO	Reserved addresses.
0x018	Interrupt Status	RW1C	Indicates the status of the interrupts.
0x01C	Interrupt Enable	RW	Indicates the interrupts that can trigger the interrupt output port.
0x020	Interrupt Set	WO	Sets the interrupt status.
0x024-0x03C	Reserved	RO	Reserved addresses.
0x040	Secondary Wire Control Status	RW	Sets the secondary wire stimulus, status and wire select for secondary wires less than 16. Note that this is generated when there is an existing secondary wire that is determined by the *Total Number of Secondary Wires* attribute.
0x044	Secondary Wire Select	RW	Sets the upper bits of the wire select. This is generated when the number of secondary wires is greater than 16. Note that this is generated when there is an existing secondary wire that is determined by the *Total Number of Secondary Wires* attribute.
0x048-0x0FC	Reserved	RO	Reserved addresses.
0x100	Target Select	RW	Sets the specific target.
0x104	User Command	RW	Sets the command from the user.
0x108	User Write Data	RW	Indicates the data to be sent to the Target.
0x10C	User Read Data	RW	Indicates the data received from the Target.
0x200-0x3FC	Reserved	RO	Reserved addresses.
Target Registers			
0x400	M-PESTI Target 0 Status	RO	Indicates the status of the M-PESTI target 0.
0x404	M-PESTI Target 0 Control Status	RW	Sets the configuration and indicates the status of the M-PESTI target 0.
0x408	M-PESTI Target 0 Virtual Wire Configuration	RW	Sets the number of virtual input and output bytes of the M-PESTI Target 0. Applicable when *Enable Programmable Virtual Wire Settings* == Checked.
0x40C	Reserved	RO	Reserved addresses for Target 0.

Offset APB	Register Name	Access Type	Description
Virtual Wire Input[1]			
0x800-0x80C	M-PESTI Target 0 Virtual Wire Input	RO	Indicates the data received from the Target via Virtual Wire.
Virtual Wire Output[1]			
0xC00-0xC0C	M-PESTI Target 0 Virtual Wire Output	RW	Indicates the data to be sent to the Target via Virtual Wire.
Payload			
0x1000-0xXXXX	Target Payload	RO	Indicates device information of each M-PESTI target.

Note:

1. Virtual Wire registers are generated based on the *Number of Virtual Wire Input Bytes* and *Number of Virtual Wire Output Bytes* attributes per target. Unused registers are considered as reserved addresses.

5.1. Configuration 0x004

Table 5.3. Configuration 0x004

Field	Name	Access	Width	Reset
[7:0]	Clock Pulse Width	RW	8	Refer to the *Calculated Clock Divider Setting* attribute.

Clock Pulse Width [7:0]

- Default – calculated divider setting based on the default *System Clock Frequency* attribute.
- Ranged from clock divider setting 2-200.

Notes:

1. Bit 0 is RO. Given that, clock pulse width configuration can only be set as multiples of 2.
2. Writing in this register changes the default clock divider setting calculated from the *System Clock Frequency* attribute and may cause improper behavior when wrongly configured. This affects the M-PESTI baud rate.

5.2. Software Reset 0x008

Table 5.4. Software Reset 0x008

Field	Name	Access	Width	Reset
[7]	No Auto Clear	RW	1	0x0
[6:2]	Reserved	RO	5	0x00
[1]	CSR Reset	RW	1	0x0
[0]	IP Core Reset	RW	1	0x0

No Auto Clear [7]

- 0 – Software reset is set to auto clear.
- 1 – Software reset value remains until it is reprogrammed.

CSR Reset [1]

Active high software reset that resets the Control, Configuration, and Status Registers to its default values aside from the Virtual Wire Input and Virtual Wire Output registers.

IP Core Reset [0]

Active high software reset that resets all blocks except the Control, Configuration, and Status Registers. This also clears target and payload registers that are controlled by the IP core.

5.3. Interrupt Status 0x018

Table 5.5. Interrupt Status 0x018

Field	Name	Access	Width	Reset
[7]	User Command Receive Timeout	RW1C	1	0x0
[6]	User Command Done	RW1C	1	0x0
[5]	User Command Received Data Available	RW1C	1	0x0
[4]	Discovery Break Assertion Timeout	RW1C	1	0x0
[3]	Good Payload Received	RW1C	1	0x0
[2]	Virtual Wire Input Updated	RW1C	1	0x0
[1]	Discovery Phase Error Detected	RW1C	1	0x0
[0]	Active Phase Error Detected	RW1C	1	0x0

User Receive Timeout [7]
- 0 – No timeout to flag.
- 1 – Initiator did not receive the expected response from target. Receive timeout is 250 ms.

User Command Done [6]
- 0 – No user command to execute or user command is not yet executed.
- 1 – Flags that the user command has been executed.

User Command Received Data Available [5]
- 0 – No updated data in the user read data register.
- 1 – An updated data is available in the user read data register.

Discovery Break Assertion Timeout [4]

This bit indicates that at least one of the targets reached the discovery break assertion timeout.
- 0 – Did not reached the discovery break assertion timeout set in the *Target Discovery Break Assertion Time* attribute.
- 1 – Reached the discovery break assertion timeout set in the *Target Discovery Break Assertion Time* attribute

Good Payload Received [3]

This bit indicates that at least one of the targets received a good payload.
- 0 – Initiator received a payload with a bad checksum or wrong parity bit from the target.
- 1 – Initiator received a payload with a good checksum from the target.

Virtual Wire Input Updated [2]

This bit indicates that at least one of the targets has an updated virtual wire input register.
- 0 – Virtual Wire Input data is not updated.
- 1 – Virtual Wire Input data is updated.

Discovery Phase Error Detected [1]

This bit indicates that at least one of the targets has a discovery phase error.
- 0 – No errors are detected from any targets.
- 1 – Error (this flags CRC error, parity error and/or framing error) is detected from target(s).

Active Phase Error Detected [0]
- 0 – No errors are detected from any targets.
- 1 – Error (this flags active phase receive timeout, parity error and/or framing error) is detected from target(s).

5.4. Interrupt Enable 0x01C

Table 5.6. Interrupt Enable 0x01C

Field	Name	Access	Width	Reset
[7]	User Command Receive Timeout Enable	RW	1	0x0
[6]	User Command Done Enable	RW	1	0x0
[5]	User Command Received Data Available Enable	RW	1	0x0
[4]	Discovery Break Assertion Timeout Enable	RW	1	0x0
[3]	Good Payload Received Enable	RW	1	0x0
[2]	Virtual Wire Input Updated Enable	RW	1	0x0
[1]	Discovery Phase Error Detected Enable	RW	1	0x0
[0]	Active Phase Error Detected Enable	RW	1	0x0

Interrupt Enable [7:0]

When enabled, the corresponding interrupt status asserts the interrupt port.

5.5. Interrupt Set 0x020

Table 5.7. Interrupt Set 0x020

Field	Name	Access	Width	Reset
[7]	User Command Receive Timeout Set	WO	1	0x0
[6]	User Command Done Set	WO	1	0x0
[5]	User Command Received Data Available Set	WO	1	0x0
[4]	Discovery Break Assertion Timeout Set	WO	1	0x0
[3]	Good Payload Received Set	WO	1	0x0
[2]	Virtual Wire Input Updated Set	WO	1	0x0
[1]	Discovery Phase Error Detected Set	WO	1	0x0
[0]	Active Phase Error Detected Set	WO	1	0x0

Interrupt Set [7:0]

This register is mainly for testing purposes.

When enabled, the corresponding interrupt status is asserted. Writing zero in this register is ignored.

5.6. Secondary Wire Control Status 0x040

Table 5.8. Secondary Wire Control Status 0x040

Field	Name	Access	Width	Reset
[7:4]	Lower Bits Wire Select	RW	4	0x0
[3]	Reserved	RO	1	0x0
[2]	Secondary Wire	RW	1	0x0
[1]	Reserved	RO	1	0x0
[0]	Secondary Wire Stimulus	RW	1	0x1

Lower Bits Wire Select [7:4]

This register is generated when the *Total Number of Secondary Wires* attribute is less than 16. This is the lower bits of the wire select from the Secondary Wire Select Register.

Secondary Wire [2]

This is generated when the *Enable CSR for Secondary Wire* attribute is checked. This reflects the value of the selected secondary wire.

Secondary Wire Stimulus [0]

This bit is an active low signal and sets the value of the selected secondary wire stimulus. This register must be cleared by writing 0x1 before sending stimulus to the next secondary wire.

Note: When Secondary Wire Control Status register is not generated. The reset value of the register is 0x00.

5.7. Secondary Wire Select 0x044

Table 5.9. Secondary Wire Select 0x044

Field	Name	Access	Width	Reset
[7:0]	Upper Bits Wire Select	RW	8	0x00

Upper Bits Wire Select [7:0]

This register is generated when the *Total Number of Secondary Wires* attribute exceeds 16. This is the upper bits of the wire select from the Secondary Wire Select Register.

5.8. Target Select 0x100

Table 5.10. Target Select 0x100

Field	Name	Access	Width	Reset
[7:6]	Reserved	RO	2	0x00
[5:0]	Target Select	RW	6	0x00

Target Select [5:0]

Selects a specific target location to send the broadcast command. This covers the maximum number of targets.

5.9. User Command 0x104

Table 5.11. User Command 0x104

Field	Name	Access	Width	Reset
[7]	Send Data	RW	1	0x0
[6]	Expect Data/ Abort	RW	1	0x0
[5]	Broadcast	RW	1	0x0
[4:0]	Reserved	RO	5	0x00

Send Data [7]

This bit auto-clears once done

- 0 – Initiator is not sending data to any target.
- 1 – Initiator is sending data to the target/s.

Expect Data/ Abort [6]

This bit auto-clears once done when Allow Multiple User Receive Bytes parameter is disabled. Otherwise, it is necessary to write 0 to clear.

When broadcast = 1,
- 0 – Wait for the current transaction to finish before proceeding to broadcast.
- 1 – Abort the current transaction then proceed to broadcast.

When broadcast = 0,
- 0 – No response is expected from the target.
- 1 – Response is expected from the target after sending the command.

Broadcast [5]
- 0 – Sends broadcast command to a single target.
- 1 – Sends broadcast command to all targets.

5.10. User Write Data 0x108

Table 5.12. User Write Data 0x108

Field	Name	Access	Width	Reset
[7:0]	User Write Data	RW	8	0x00

User Write Data [7:0]

When send data (User Command [7] register) is set to 1, this register stores the data to be transmitted to the target/s.

5.11. User Read Data 0x10C

Table 5.13. User Read Data 0x10C

Field	Name	Access	Width	Reset
[7:0]	User Read Data	RO	8	0x00

User Read Data [7:0]

This register stores the data received from the target.

5.12. M-PESTI Target N Status 0x400+(N*16)

Table 5.14. M-PESTI Target N Status 0x400+(N*16)

Field	Name	Access	Width	Reset
[7]	CRC Error	RW1C	1	0x0
[6]	Parity Error	RW1C	1	0x0
[5]	Active Phase Receive Timeout	RW1C	1	0x0
[4]	Active Phase Error	RW1C	1	0x0
[3:2]	Current Phase	RO	2	0x0
[1:0]	Discovery Status	RO	2	0x0

Note:

* N is the target number.

CRC Error [7]
- 0 – No discovery phase CRC error is detected.
- 1 – Discovery phase CRC error is detected.

Active phase CRC-8 is not covered.

Note: This becomes sticky once Discovery Retry Rollover asserted (three unsuccessful retries has been reached).

Parity Error [6]
- 0 – No parity error is detected.
- 1 – Parity error is detected.

Note: This becomes sticky once Discovery Retry Rollover asserted (three unsuccessful retries has been reached).

Active Phase Receive Timeout [5]
- 0 – No active phase received timeout error is detected.
- 1 – Active phase received timeout error is detected.

Active Phase Error [4]
- 0 – No active phase error is detected.
- 1 – Active phase error is detected (includes receive timeout, parity error, and framing error).

Current Phase [3:2]
- 00 – Reset.
- 01 – UNUSED.
- 10 – Discovery phase.
- 11 – Active phase.

Discovery Status [1:0]
- 00 – Absent.
- 01 – Present.
- 10 – M-PESTI is present with good payload.
- 11 – M-PESTI is present without successful payload yet.

5.13. M-PESTI Target N Control Status 0x404+(N*16)

Table 5.15. M-PESTI Target N Control Status 0x404+(N*16)

Field	Name	Access	Width	Reset
[7]	Reserved	RO	1	0x0
[6]	Discovery Retry Rollover	RW1C	1	0x0
[5]	Discovery Phase Error	RW1C	1	0x0
[4]	Discovery Break Assertion Timeout	RW1C	1	0x0
[3:2]	Reserved	RO	2	0x0
[1]	Active Phase Enable	RW	1	Refer to the Active Phase Enable attribute.
[0]	Discovery Phase Enable	RW	1	Refer to the Discovery Phase Enable attribute.

Note:
* N is the target number.

Discovery Retry Rollover [6]
- 0 – Initiator has not reached the three discovery retries.
- 1 – Initiator has attempted discovery for three unsuccessful retries (initial + two retries).

Discovery Phase Error [5]
- 0 – No discovery phase error.
- 1 – Discovery phase error has been flagged, either parity, framing, or CRC error.

Note: This becomes sticky once Discovery Retry Rollover asserted (three unsuccessful retries has been reached).

Discovery Break Assertion Timeout [4]
- 0 – No discovery break assertion timeout.
- 1 – Discovery break assertion timeout is flagged when break is held for longer than the set assertion time in *Target Discovery Break Assertion Time* attribute.

Active Phase Enable [1]
- 0 – Active phase is disabled.
- 1 – Active phase is enabled.

Discovery Phase Enable [0]
- 0 – Discovery phase is disabled.
- 1 – Discovery phase is enabled.

5.14. M-PESTI Target N Virtual Wire Configuration 0x408+(N*16)

Table 5.16. M-PESTI Target N Virtual Wire Configuration 0x408+(N*16)

Field	Name	Access	Width	Reset
[12:8]	Number of Virtual Wire Output Bytes	RW	5	Refer to the *Number of Virtual Wire Output Bytes* attribute.
[7:5]	Reserved	RO	3	0x00.
[4:0]	Number of Virtual Wire Input Bytes	RW	5	Refer to the *Number of Virtual Wire Input Bytes* attribute.

Note:
* N is the target number.
Allowed reconfiguration values must be less than or equal to the set Number of Virtual Wire Output Bytes and Number of Virtual Wire Output Bytes attributes.

5.15. M-PESTI Target N Virtual Wire Input 0x800+(N*16)

Table 5.17. M-PESTI Target N Virtual Wire Input 0x800+(N*16)

Field	Name	Access	Width	Reset
[7:0]	Virtual Wire Input	RO	32	0x00

Note:
* N is the target number.

Virtual Wire Input [7:0]

This register is not cleared by any of the software resets.

This register stores the data received from the target.

5.16. M-PESTI Target N Virtual Wire Output 0xC00+(N*16)

Table 5.18. M-PESTI Target N Virtual Wire Output 0xC00+(N*16)

Field	Name	Access	Width	Reset
[7:0]	Virtual Wire Output	if Enable Virtual Wire as Ports, RO, else, RW	8	0x00

Note:
* N is the target number

Virtual Wire Output [7:0]

This register is not cleared by any of the software resets.

This register stores the data sent to the target.

6. Example Design

The M-PESTI Initiator IP example design allows you to compile, simulate, and test the M-PESTI Initiator IP on the following Lattice evaluation boards:

- MachXO5-NX Development Board

6.1. Example Design Supported Configuration

Note: In the table below, ✓ refers to a checked option in the M-PESTI Initiator IP core example design and an emdash (—) refers to an unchecked option or a non-applicable option in the M-PESTI Initiator IP core example design.

Table 6.1. M-PESTI Initiator IP Configuration Supported by the Example Design

M-PESTI Initiator IP GUI Parameter	M-PESTI Initiator IP Configuration Supported in the Example Demo Design
System Clock Frequency (MHz)	25
System Clock Period (ns)	40 (Display only)
M-PESTI Baud Rate (kHz)	250 (Display only)
System Clock Period (ns)	4,000 (Display only)
Calculated Clock Divider Setting	50 (Display only)
APB Data Width	32 (Display only)
Target Discovery Break Assertion Time (us)	1,000
Number of Target Devices	2
Total Number of Secondary Wires	0
Maximum Static Discovery Payload Size	64
Discovery Phase Enable	✓
Active Phase Enable	✓
Enable CSR for Secondary Wire	—
Enable Glitch Filter	—
Allow Multiple User Receive Bytes	✓
Maximum User Receive Byte Size	2
Payload Memory Allocation Option (for single target only)	RAM
Payload Memory Allocation	RAM (Display only)
Include I/O Primitive	✓
Enable Programmable Virtual Wire	✓
Enable Virtual Wire as Ports (in hex)	3
Target 0	
Number of Virtual Wire Input Bytes	1
Number of Virtual Wire Output Bytes	1
Target 1	
Number of Virtual Wire Input Bytes	2
Number of Virtual Wire Output Bytes	2

6.2. Overview of the Example Design and Features

The example design discussed in this section is created using the RISC-V MC SoC project template in the Lattice Propel Development Suite. The generated project includes the following components:

- Processor – RISC-V MC w/ PIC/TIMER
- GPIO
- Asynchronous SRAM
- UART – Serial port
- Phase-locked loop (PLL)
- Glue logic

M-PESTI Initiator and M-PESTI Targets are instantiated and connected in the project as shown in the following figure. In this example design, both initiator and targets are instantiated in the same system. In actual hardware or use case, the M-PESTI Initiator IP core can be connected to external target devices.

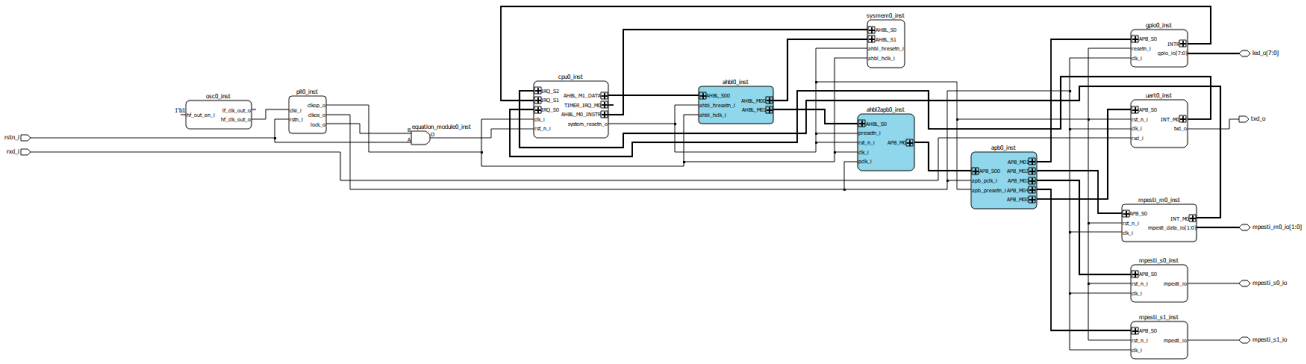

Figure 6.1. M-PESTI Initiator IP Core in an SoC Project

An Embedded C/C++ project is also created in the Propel software to enable the development and debugging application code for different IP features. The M-PESTI Initiator IP core features can be tested by initiating discovery and active phase transactions and sending user commands. Runtime configuration of IP core and feature testing can be done through C-Code Test Routine. The following figure shows an example routine for reading the M-PESTI discovery payload.

```
unsigned int mpesti_mst_rd_disc_payload(mpesti_mst_handle_t *handle, unsigned int *data_buf)
{
  unsigned int status;
  int          cnt;

  if(handle->base_addr != ZERO)
  {
    for(cnt=ZERO; cnt<handle->disc_payload_size; cnt++)
    {
      *(unsigned int *)(data_buf + cnt) = *(volatile unsigned int *)(handle->base_addr + MSTADR_INTMEM_BASE + cnt);
    }
    status=SUCCESS;
  }
  else
  {
    status=FAILURE;
  }
  return status;
}
```

Figure 6.2. Sample C Code Test Routine

6.3. Example Design Components

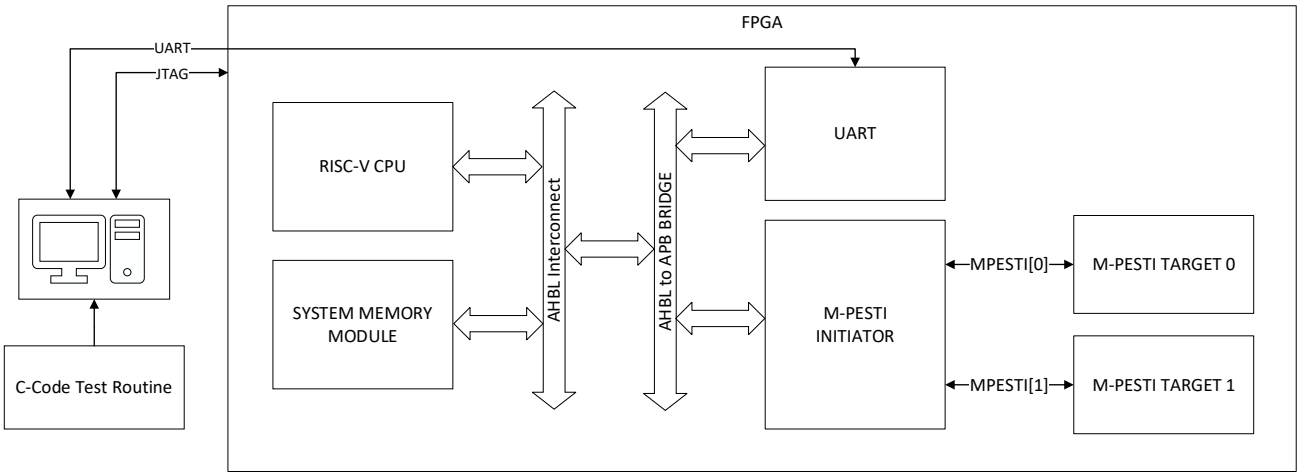

Figure 6.3. M-PESTI Initiator IP Example Design Block Diagram

The M-PESTI Initiator example design includes the following blocks:

- RISC-V CPU – Passes the C-Code Test Routing from system memory to system bus and handles interrupts.
- Memory module – Contains command to be done for testing.
- System bus – AHB-Lite systems bus for transfers between memory and IP.
- M-PESTI Initiator IP.
- M-PESTI target devices.

6.4. Generating the Example Design

To generate the example design, follow these steps:

1. Launch the Lattice Propel software and set your workspace directory.
2. In the Propel software, create a new Lattice SoC Design Project. Click **File > New > Lattice SoC Design Project**.
3. The Create SoC Project window opens.

   In **Template Design**, select **RISC-V MC SoC Project**. Click **Next**. In **Device Select** section, indicate the correct details of the device or board that you will use. In the following figure, the device is set to **LFMXO5-25-9BBG400C** because the MachXO5-NX Development Board is used in hardware testing. Click **Next**.
4. Click **Finish**.

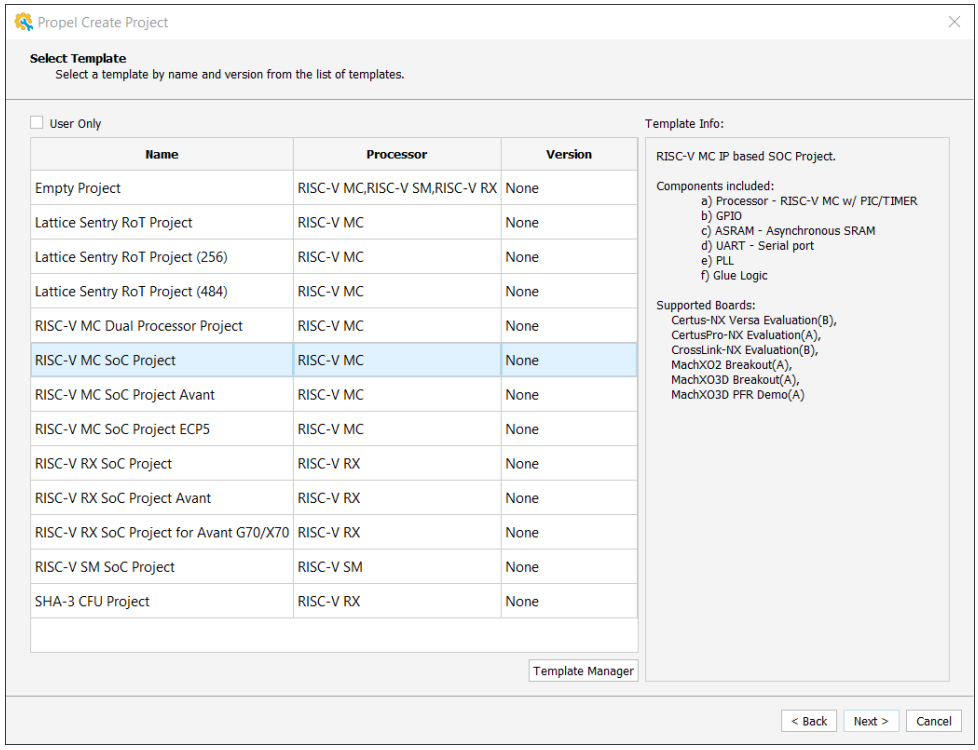

Figure 6.4. Select Template

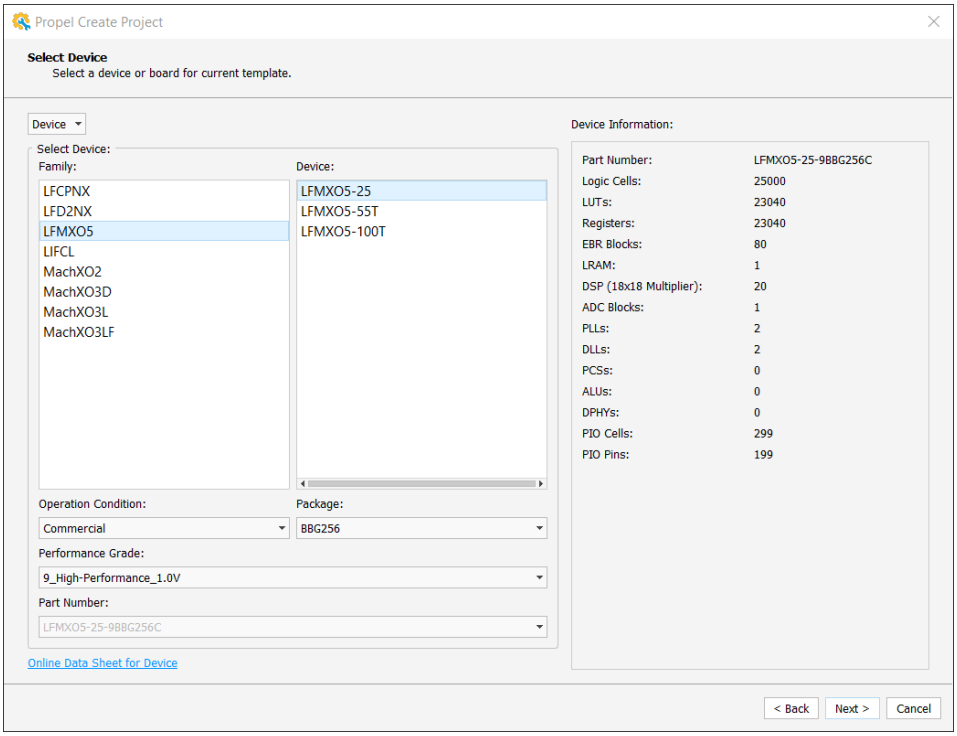

Figure 6.5. Create SoC Project

5. Run the Propel Builder by clicking on the icon or **LatticeTools > Open Design** in the Propel Builder. The Propel Builder opens and loads the design template.
6. In the **IP Catalog** tab, instantiate the M-PESTI Initiator IP core. For more information, refer to the Generating and Instantiating the IP section.
7. After generating the IP core, the **Define Instance** window opens. Modify the instance name when needed then click **OK**.

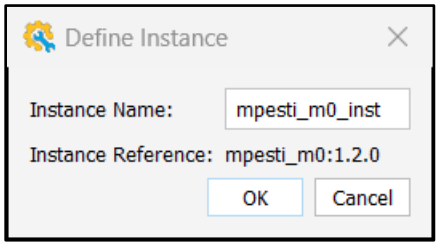

Figure 6.6. Define Instance

8. Instantiate M-PESTI targets as well. You own target must be provided. For more information, refer to the Generating and Instantiating the IP section.
9. Connect the instantiated IP cores to the system. Refer to Figure 6.1 for the connections used in this IP. You must update other components of the system for clock and reset sources, interrupt, and bus interface.
10. Click the icon or **Design > Run Radiant** to launch the Lattice Radiant software.
11. Update your constraints file accordingly and generate the programming file.
12. In the Lattice Propel software, build your SoC project to generate the system environment needed for the embedded C/C++ project. Select your SoC project, then click **Project > Build Project**.
13. Check the build result from the **Console** view.
14. Generate a new Lattice C/C++ project by clicking on **File > New > Lattice C/C++ Project**. Update the project name and click **Next > Finish**.

For more information on using the Propel software, refer to the A Step-By-Step Approach to Lattice Propel Application Note (FPGA-AN-02052).

6.5. Simulating the Example Design

The Lattice Propel Builder software also has a verification project mode that can be used to generate a simulation environment. The procedure for generating a verification project for the M-PESTI Initiator IP core is described in the following section.

To simulate the example design, follow these steps:

1. From the SoC Project, perform the pre-simulation requirements enumerated in the Verification Project Flow section of the A Step-By-Step Approach to Lattice Propel Application Note (FPGA-AN-02052).
2. Click on the Switch Verification and SoC design icon to switch to verification project.
3. Reload dut_inst by double clicking on it. The **Reload sbx** window displays, as shown in the following figure. Click **Yes** to continue.

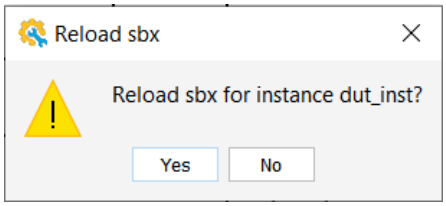

Figure 6.7. Reload sbx

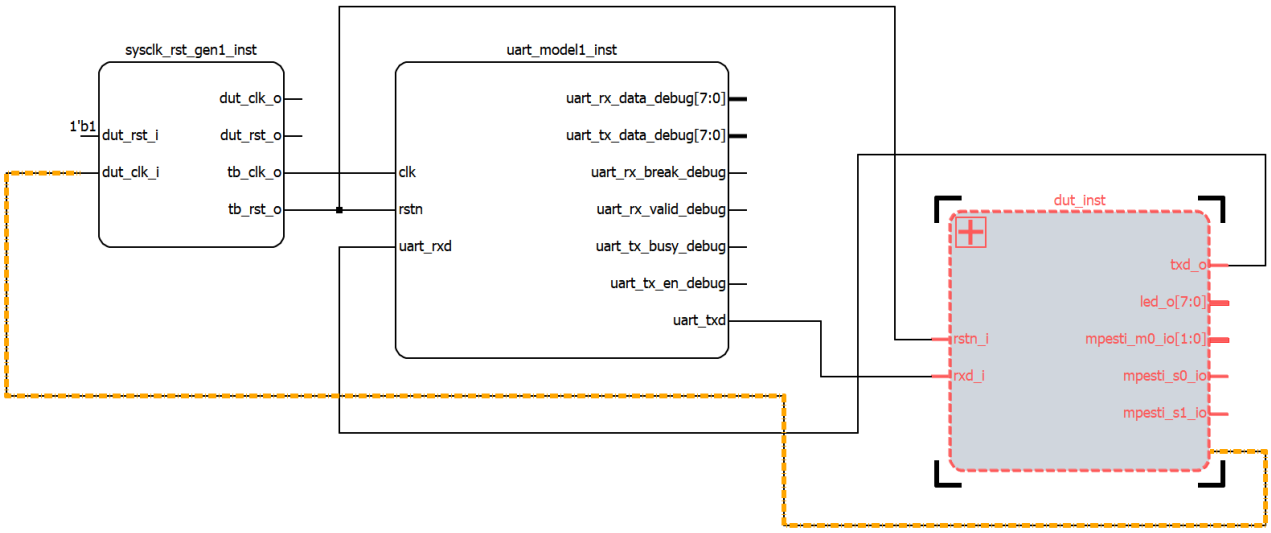

Figure 6.8. Verification Project Schematic

4. Click the Generate icon  to generate the simulation environment. The testbench file includes scripts for the chosen simulator, file lists, and other files are generated in the verification folder of the SoC project.
5. To see the full IP simulation behavior for this example design, connect the M-PESTI model to the IP instance in the generated testbench file as shown in the following figure and add the M-PESTI simulation model in the generated file list.

```
/*----------------------Instantiation Blocks----------------------*/
/* This section cover the instantiation of DUT, VIPs, Simulation  */
/* Models, and other IP/Components.                               */
/*----------------------------------------------------------------*/


mpesti_soc
dut_inst
(
  .mpesti_s0_io( ),
  .mpesti_s1_io( ),
  .rstn_i(sysclk_rst_gen1_inst_tb_rst_o_net),
  .rxd_i(uart_model1_inst_uart_txd_net),
  .txd_o(uart_model1_inst_uart_rxd_net),
  .led_o( ),
  .mpesti_m0_io( )
);

initial begin
  sysclk_rst_gen1_inst.SysClk_Rst_GenMon_inst.set_clk_freq_mhz(50);
  sysclk_rst_gen1_inst.SysClk_Rst_GenMon_inst.start_clk();
end

sysclk_rst_gen1
sysclk_rst_gen1_inst
(
  .dut_clk_i(sysclk_rst_gen1_inst_dut_clk_i_net),
  .dut_rst_i(32'd1),
  .tb_clk_o(sysclk_rst_gen1_inst_tb_clk_o_net),
  .tb_rst_o(sysclk_rst_gen1_inst_tb_rst_o_net)
);

uart_model1
uart_model1_inst
(
  .clk(sysclk_rst_gen1_inst_tb_clk_o_net),
  .rstn(sysclk_rst_gen1_inst_tb_rst_o_net),
  .uart_rxd(uart_model1_inst_uart_rxd_net),
  .uart_txd(uart_model1_inst_uart_txd_net)
);

endmodule
```

Figure 6.9. SoC and Simulation Model Instantiation

6. Click the Launch Simulation icon  to run the simulation.

6.6. Hardware Testing

The generated bitstream file from the procedure in the Generating the Example Design section is downloaded to the MachXO5-NX Development Board via the Radiant programmer. The Reveal analyzer is added to the Radiant software project to verify the output behavior of the IP core.

7. Designing with the IP

This section provides information on how to generate the IP core using the Lattice Radiant software and how to run simulation and synthesis. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software User Guide.

7.1. Generating and Instantiating the IP

You can use the Lattice Radiant software to generate IP modules and integrate them into the device architecture. The following steps describe how to generate the M-PESTI Initiator IP core in the Lattice Radiant software:

1. Create a new Lattice Radiant software project or open an existing project.
2. In the **IP Catalog** tab, double-click **M-PESTI Initiator** under **IP, Processors_Controllers_and_Peripherals** category. The **Module/IP Block Wizard** opens as shown in the following figure. Enter values in the **Component name** and the **Create in** fields and click **Next**.

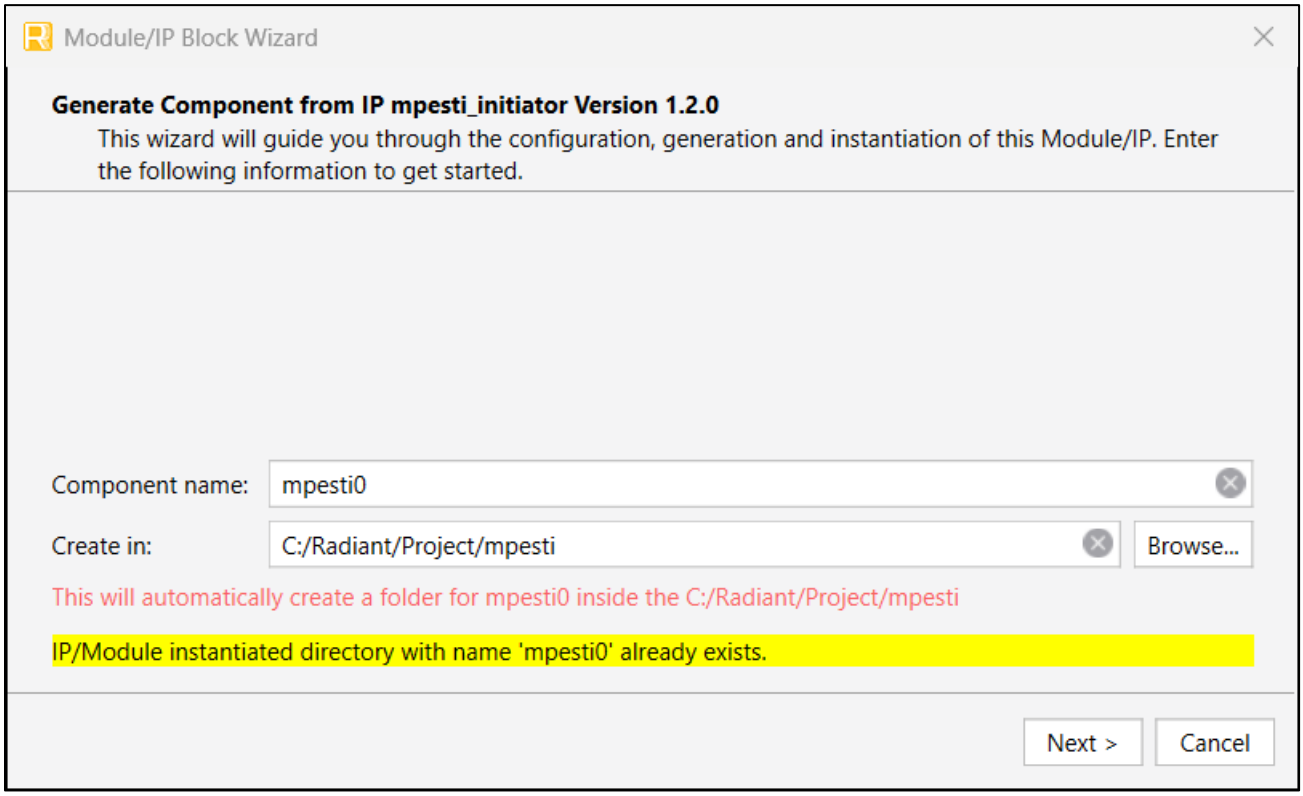

Figure 7.1. Module/IP Block Wizard

3. In the next **Module/IP Block Wizard** window, customize the selected M-PESTI Initiator IP core using the drop-down lists and check boxes. For details on the configuration options, refer to the IP Parameter Description section.

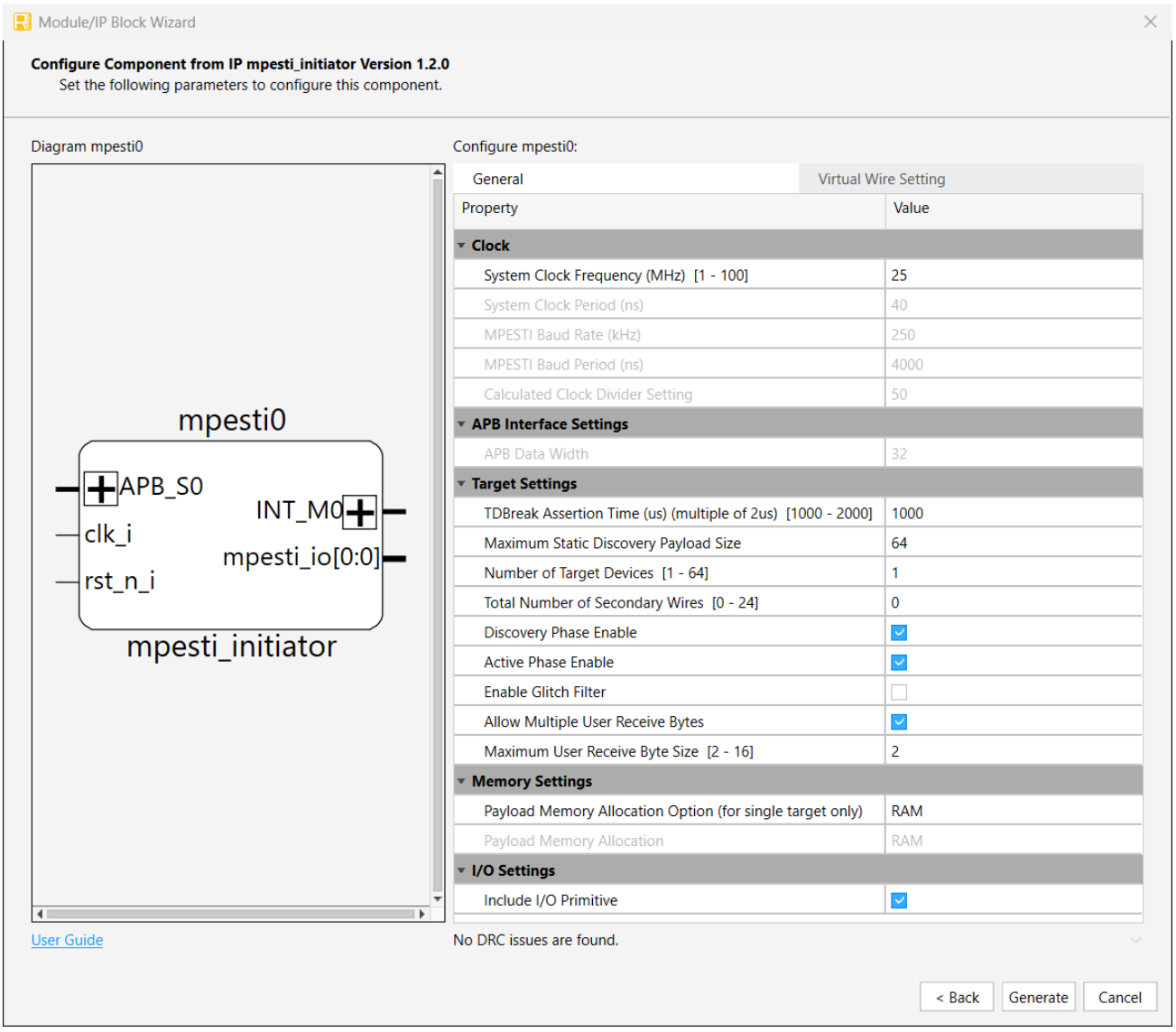

Figure 7.2. M-PESTI Initiator IP Core Configuration Example

4. Click **Generate**. The **Check Generated Result** dialog box opens. The following figure shows the design block messages and results.

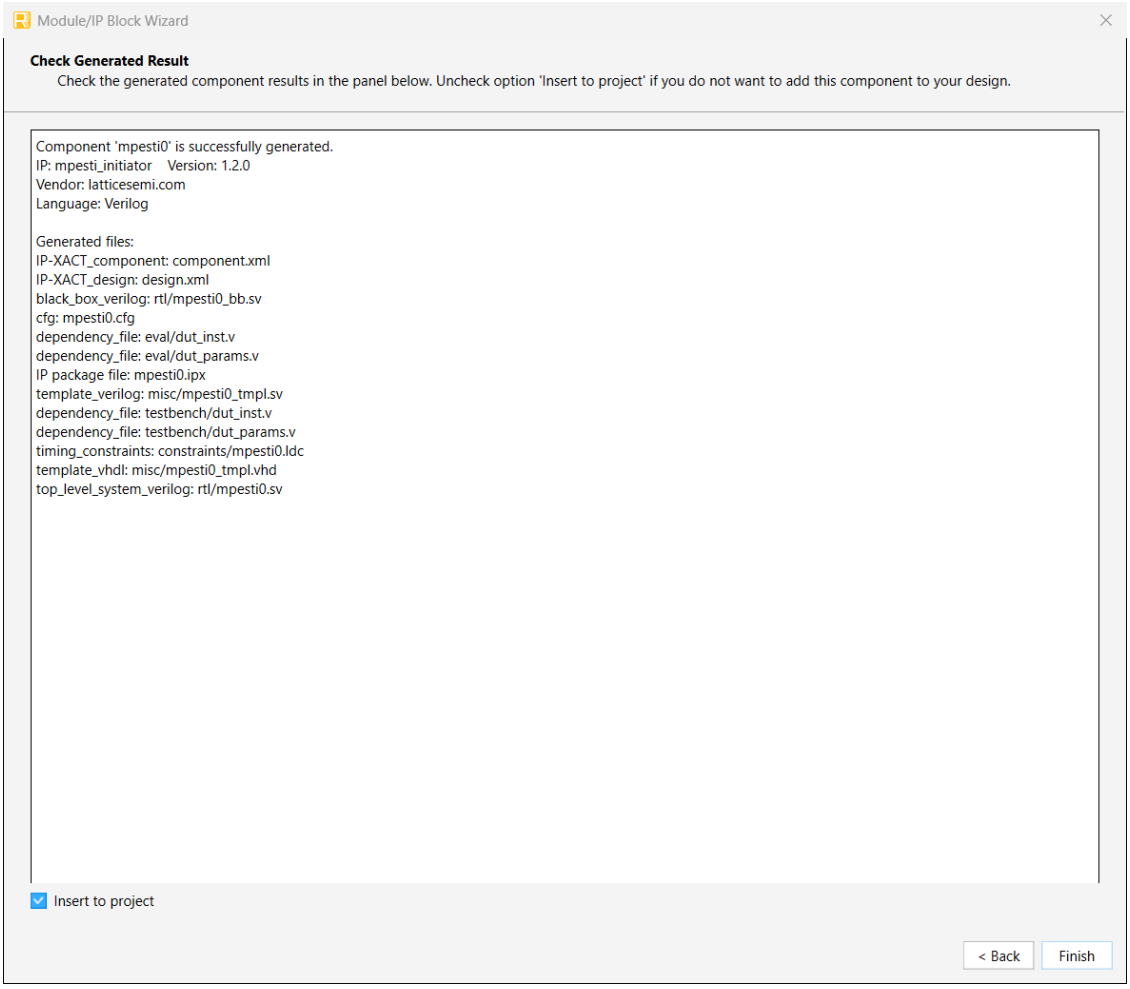

Figure 7.3. Check Generated Result

5. Click **Finish**. All the generated files are placed under the directory paths in the **Create in** and the **Component name** fields.

7.1.1. Generated Files and File Structure

The generated M-PESTI Initiator IP module package includes the closed-box (*<Component name>_bb.v*) and instance templates (*<Component name>_tmpl.v/vhd*) that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file (*<Component name>.v*) that can be used as an instantiation template for the module is also provided. You may also use this example as the starting template for your top-level design.

Table 7.1. Generated File List

Generated File	Description
<Component name>.ipx	This file contains the information on the files associated to the generated IP.
<Component name>.cfg	This file contains the parameter values used in IP configuration.
component.xml	Contains the ipxact: component information of the IP.
design.xml	Documents the configuration parameters of the IP in IP-XACT 2014 format.
rtl/<Component name>.v	This file provides an example RTL top file that instantiates the module.
rtl/<Component name>_bb.v	This file provides the synthesis closed-box.
misc/<Component name>_tmpl.v *misc /<Component name>_tmpl.vhd*	These files provide instance templates for the module.

7.2. Design Implementation

Completing your design includes additional steps to specify analog properties, pin assignments, and timing and physical constraints. You can add and edit the constraints using the Device Constraint Editor or by manually creating a PDC file.

Post-Synthesis constraint files (*.pdc*) contain both timing and non-timing *constraint.pdc* source files for storing logical timing/physical constraints. Constraints that are added using the Device Constraint Editor are saved to the active *.pdc* file. The active post-synthesis design constraint file is then used as input for post-synthesis processes.

Refer to the relevant sections in the Lattice Radiant Software User Guide for more information on how to create or edit constraints and how to use the Device Constraint Editor.

7.3. Timing Constraints

You must provide proper timing and physical design constraints to ensure that your design meets the desired performance goals on the FPGA. Add the content of the following IP constraint file to your design constraints: *<IP_Instance_Path>/<IP_Instance_Name>/eval/constraint.pdc.*

The constraint file has been verified during IP evaluation with the IP instantiated directly in the top-level module. You can modify the constraints in this file with thorough understanding of the effect of each constraint.

To use this constraint file, copy the content of *constraint.pdc* to the top-level design constraint for post-synthesis.

For more information on how to constrain your design, refer to the Lattice Radiant Timing Constraints Methodology Application Note (FPGA-AN-02059).

7.4. Running Functional Simulation

You can run functional simulation after the IP core is generated.

To run the functional simulation, follow these steps:

1. Click the  icon located on the **Toolbar** to initiate the **Simulation Wizard**.

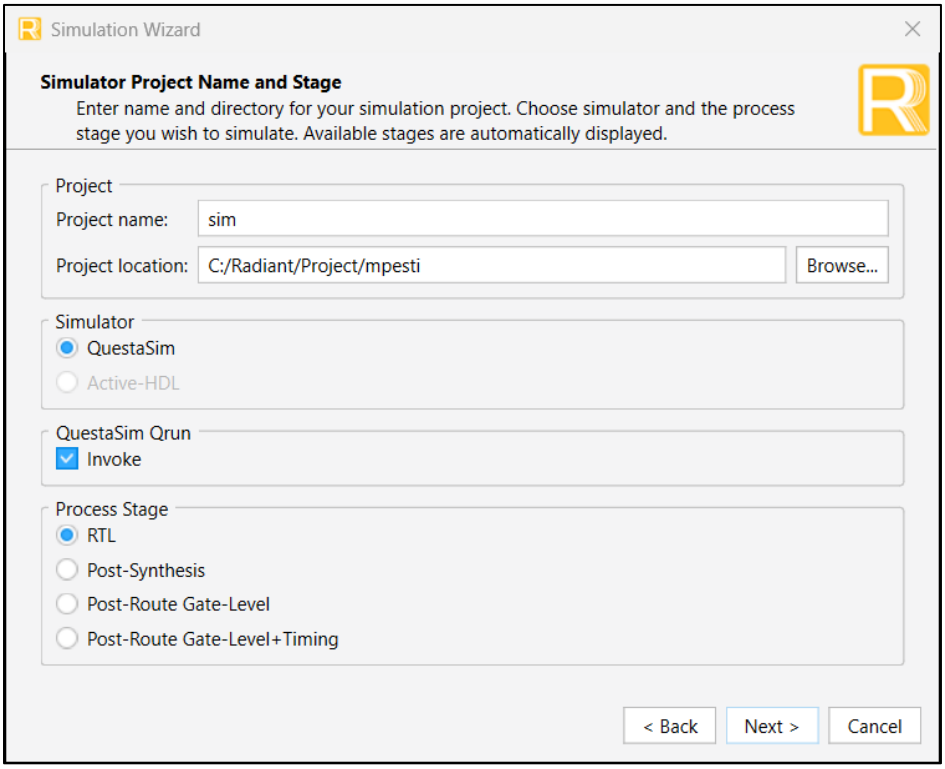

Figure 7.4. Simulation Wizard

2. Click **Next** to open the **Add and Reorder Source** window.

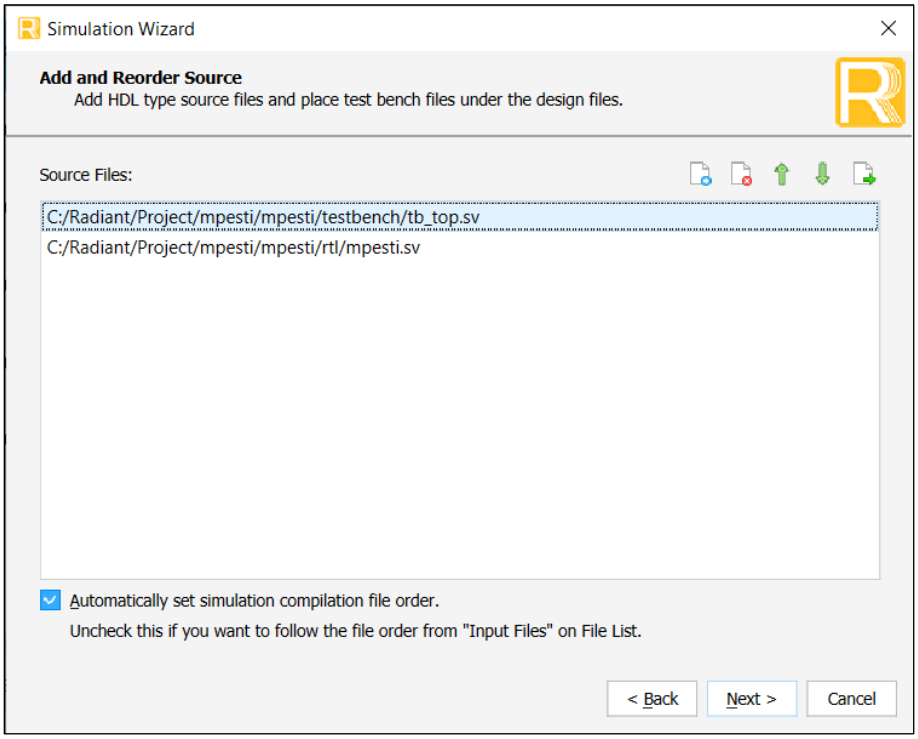

Figure 7.5. Add and Reorder Source

3. Click **Next**. The **Summary** window displays.

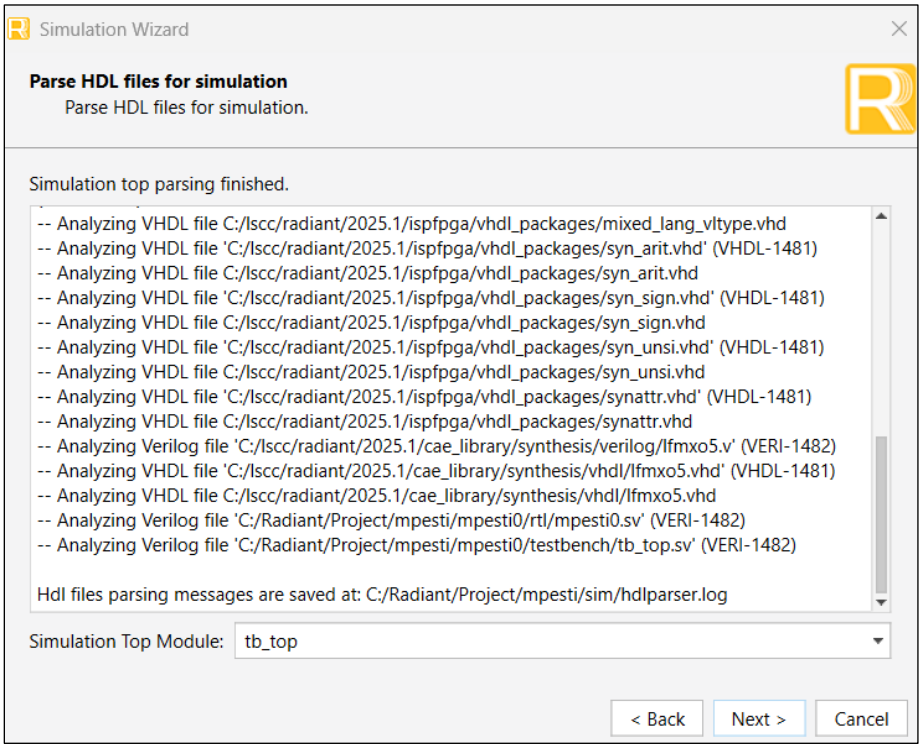

Figure 7.6. Parse HDL Files for Simulation

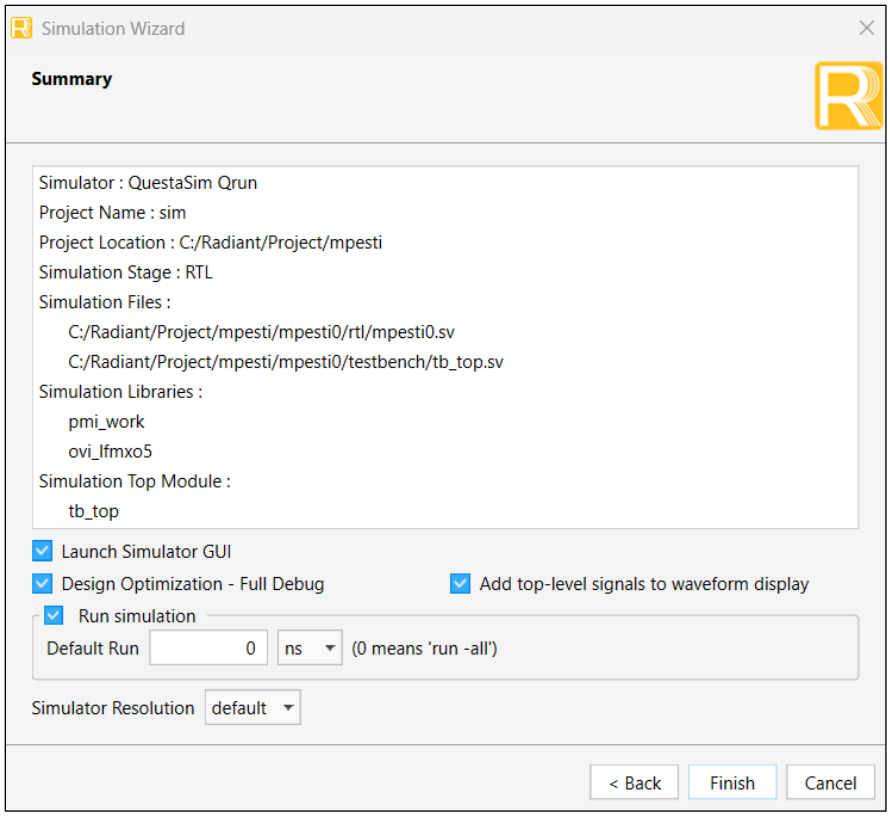

Figure 7.7. Summary Window

4. Click **Finish** to run the simulation.

The following figures show the transcript and waveform of the simulation result.

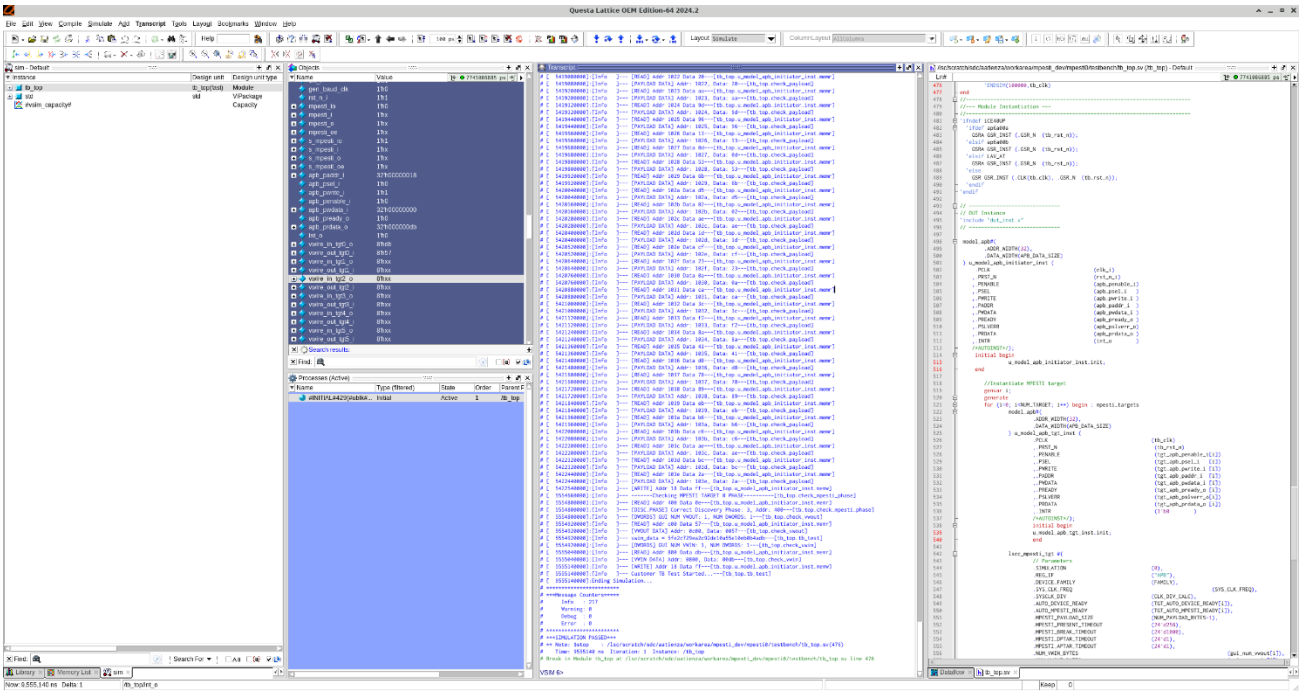

Figure 7.8. Transcript of the Simulation Result

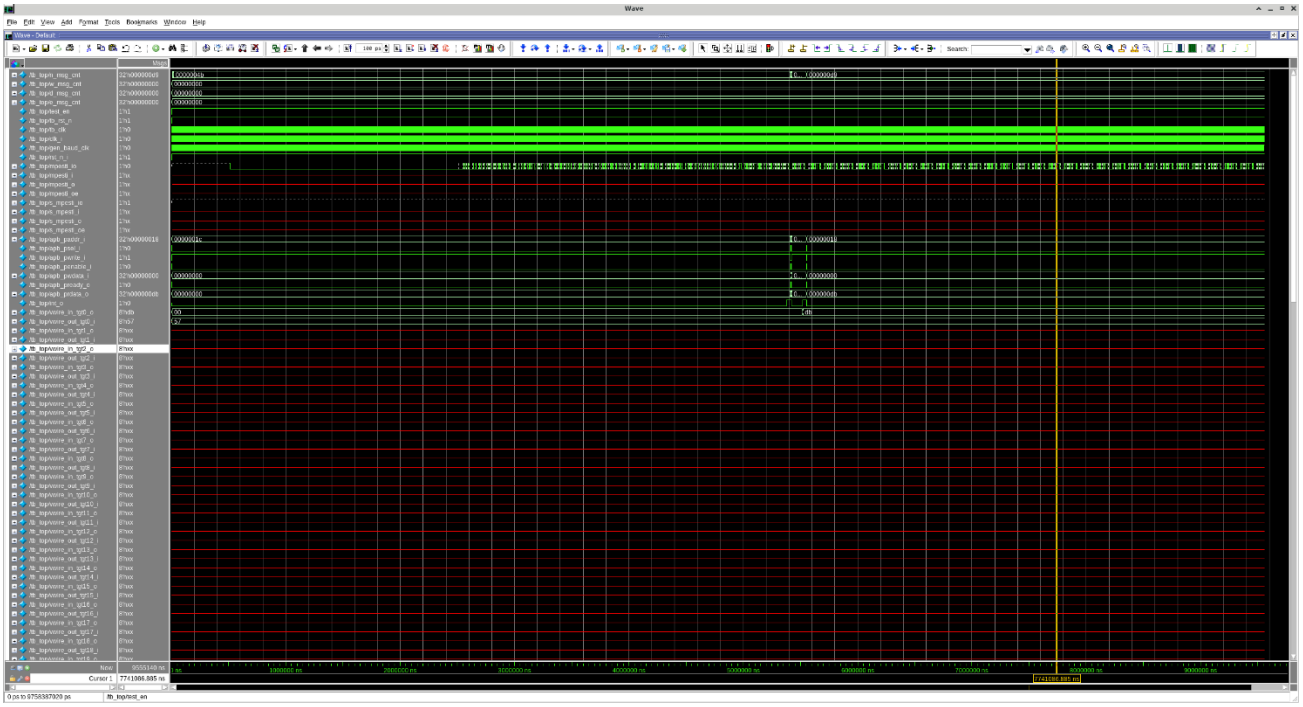

Figure 7.9. Simulation Waveform

Appendix A. Resource Utilization

The following table shows a sample resource utilization of the M-PESTI Initiator IP core v1.1.0 using the LFMXO5-25-7BBG400I device with the Synplify Pro synthesis tool in the Lattice Radiant software 2024.2. Default configuration is used and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.1. Resource Utilization for LFMXO5-25-7BBG400I Device (IP v1.1.0)

IP Configuration	clk_i Fmax (MHz)[1]	Registers	LUTs[2]	EBRs	DSPs
Default configuration Number of Payload Bytes = 16	128.584	366	592	1	0
Memory Allocation = Register	128.386	417	629	1	0
Number of Payload Bytes = 64 Number of Targets = 16	130.429	1,419	1,925	1	0
Number of Payload Bytes = 64 Number of Targets = 64	96.927	4,757	6,431	1	0

Notes:

1. Fmax is generated when the FPGA design only contains the M-PESTI Initiator module, and the target frequency is 25 MHz. These values may be reduced when user logic is added to the FPGA design.
2. The distributed RAM utilization is accounted for in the total LUT4s utilization. The actual LUT4 utilization is distribution among logic, and ripple logic.

The following table shows a sample resource utilization of the M-PESTI Initiator IP core v1.2.0 using the LFMXO5-25-7BBG400I device with the Synplify Pro synthesis tool in the Lattice Radiant software 2025.1. Default configuration is used and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.2. Resource Utilization for LFMXO5-25-7BBG400I Device (IP v1.2.0)

IP Configuration	clk_i Fmax (MHz)[1]	Registers	LUTs[2]	EBRs	DSPs
Default configuration	150.240	956	1,192	1	0
Number of Payload Bytes = 64 Number of Targets = 16	93.153	7,702	7,080	1	0
Number of Payload Bytes = 64 Number of Targets = 64	125.854	29,217	25,045	1	0

Notes:

1. Fmax is generated when the FPGA design only contains the M-PESTI Initiator module, and the target frequency is 25 MHz. These values may be reduced when user logic is added to the FPGA design.
2. The distributed RAM utilization is accounted for in the total LUT4s utilization. The actual LUT4 utilization is distribution among logic, and ripple logic.

The following table shows a sample resource utilization of the M-PESTI Initiator IP core v1.2.0 using the LAV-AT-E70-1LFG676I device with the Synplify Pro synthesis tool in the Lattice Radiant software 2025.1. Default configuration is used and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.3. Resource Utilization for LAV-AT-E70-1LFG676I Device (IP v1.2.0)

IP Configuration	clk_i Fmax (MHz)[1]	Registers	LUTs[2]	EBRs	DSPs
Default configuration	228.154	966	1,235	1	0
Number of Payload Bytes = 64 Number of Targets = 16	145.836	7,716	7,082	1	0
Number of Payload Bytes = 64 Number of Targets = 64	100.351	29,229	24,950	1	0

Notes:

1. Fmax is generated when the FPGA design only contains the M-PESTI Initiator module, and the target frequency is 25 MHz. These values may be reduced when user logic is added to the FPGA design.
2. The distributed RAM utilization is accounted for in the total LUT4s utilization. The actual LUT4 utilization is distribution among logic, and ripple logic.

The following table shows a sample resource utilization of the M-PESTI Initiator IP core v1.2.0 using the LN2-CT-20ES-1ASG410I device with the Synplify Pro synthesis tool in the Lattice Radiant software 2025.1. Default configuration is used and some attributes are changed from the default value to show the effect on the resource utilization.

Table A.4. Resource Utilization for LN2-CT-20ES-1ASG410I Device (IP v1.2.0)

IP Configuration	clk_i Fmax (MHz)[1]	Registers	LUTs[2]	EBRs	DSPs
Default configuration	212.675	966	1,235	1	0
Number of Payload Bytes = 64 Number of Targets = 16	150.648	7,715	7,082	1	0
Number of Payload Bytes = 64 Number of Targets = 64	113.727	29,229	24,950	1	0

Notes:

1. Fmax is generated when the FPGA design only contains the M-PESTI Initiator module, and the target frequency is 25 MHz. These values may be reduced when user logic is added to the FPGA design.
2. The distributed RAM utilization is accounted for in the total LUT4s utilization. The actual LUT4 utilization is distribution among logic, and ripple logic.

References

- M-PESTI Initiator IP Release Notes (FPGA-RN-02005)
- M-PESTI Initiator Driver API Reference (FPGA-TN-02413)
- Modular-Peripheral Sideband Tunneling Interface (M-PESTI) Base Specification
- A Step-By-Step Approach to Lattice Propel Application Note (FPGA-AN-02052)
- Lattice Radiant Timing Constraints Methodology Application Note (FPGA-AN-02059)
- M-PESTI Initiator IP web page
- Avant-E web page
- Avant-G web page
- Avant-X web page
- MachXO5-NX web page
- MachXO3D web page
- MachXO3 web page
- Mach-NX web page
- Lattice Radiant Software web page
- Lattice Propel Design Environment web page
- Lattice Solutions IP Cores web page
- Lattice Solutions Reference Designs web page
- Lattice Solutions Boards web page
- Lattice Solutions Demonstrations web page
- 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 1.3, IP v1.3.0, July 2025

Section	Change Summary

Introduction	Updated Table 1.1. Summary of the M-PESTI Initiator IP:
	• IP Requirements:
	  • Changed *FPGA Families Supported* to *Supported Devices*.
	  • Updated list of *Supported Devices*.
	• Resource Utilization:
	  • Added *Resources* and linked to Appendix A. Resource Utilization.
	• Lattice Implementation:
	  • Updated IP and Lattice Radiant Software version.

References	Added M-PESTI Initiator Driver API Reference (FPGA-TN-02413).

Revision 1.2, IP v1.2.0, June 2025

Section	Change Summary
Introduction	• Updated the IP version in Table 1.1. Summary of the M-PESTI Initiator IP. • Updated the name of the license type in Table 1.3. Ordering Part Number from *Multi-Site Perpetual* to *Single Seat Perpetual*.
Functional Description	• Updated Figure 2.1. Lattice M-PESTI Initiator IP Core Block Diagram. • Updated Table 2.1. User Interfaces and Supported Protocols. • Updated the following sentence in the Active Phase section: *Current autonomous virtual wire exchange transacts virtual wire input and output data based on the Number of Virtual Wire Input and Output Bytes attribute per target.* • Added the Target Setting Reconfiguration section. • Updated the Active Phase section. • Updated Figure 2.7. Abort Mechanism.
IP Parameter Description	Updated Table 3.1. General Attributes.
Signal Description	Updated Table 4.1. Ports Description.
Register Description	• Updated the following sentence in this section: *For example, four targets with 32 bytes payload each = 128 bytes RAM.* • Added Table 5.1. Register Access Types. • Updated Table 5.2. Register Address Map. • Updated *VWIN* to *Virtual Wire Input data* in the Interrupt Status 0x018 section. • Updated the following sentence in the Interrupt Status 0x018 section: *1 – Error (this flags active phase receive timeout, parity error and/or framing error) is detected from target(s).* • Updated Table 5.6. Interrupt Enable 0x01C. • Updated Table 5.11. User Command 0x104. • Updated the M-PESTI Target N Control Status 0x404+(N*16) section. • Added the M-PESTI Target N Virtual Wire Configuration 0x408+(N*16) section. • Updated the M-PESTI Target N Virtual Wire Input 0x800+(N*16) section. • Updated the M-PESTI Target N Virtual Wire Output 0xC00+(N*16) section.
Example Design	• Updated Table 6.1. M-PESTI Initiator IP Configuration Supported by the Example Design. • Updated Figure 6.6. Define Instance.

Designing with the IP	• Updated the following figures:
	  • Figure 7.1. Module/IP Block Wizard.
	  • Figure 7.2. M-PESTI Initiator IP Core Configuration Example.
	  • Figure 7.3. Check Generated Result.
	  • Figure 7.4. Simulation Wizard.
	  • Figure 7.9. Simulation Waveform.
	• Added the following figures:

Section	Change Summary
	• Figure 7.6. Parse HDL Files for Simulation. • Figure 7.7. Summary Window. • Figure 7.8. Transcript of the Simulation Result.

Appendix A. Resource Utilization	• Updated Table A.1. Resource Utilization for LFMXO5-25-7BBG400I Device (IP v1.1.0).
	• Added the following tables:
	  • Table A.2. Resource Utilization for LFMXO5-25-7BBG400I Device (IP v1.2.0).
	  • Table A.3. Resource Utilization for LAV-AT-E70-1LFG676I Device (IP v1.2.0).
	  • Table A.4. Resource Utilization for LN2-CT-20ES-1ASG410I Device (IP v1.2.0).

Revision 1.1, IP v1.1.0, December 2024

Section	Change Summary
All	Added IP version on the cover page.
Introduction	• Updated Table 1.1. Summary of the M-PESTI Initiator IP. • Added the IP Support Summary section. • Updated the Features section. • Added new ordering part numbers for MachXO3D and Avant devices in Table 1.3. Ordering Part Number. • Updated a column title in Table 1.3. Ordering Part Number from Single Machine to Single Seat. • Changed the section title from IP Validation Summary to Hardware Support and updated the content in the Hardware Support section.

Functional Description	• Updated Figure 2.1. Lattice M-PESTI Initiator IP Core Block Diagram
	• Updated the following sections:
	  • Clocking section
	  • Reset section
	• Updated the following sub-sections in the M-PESTI Protocol section:
	  • M-PESTI UART Frame section
	  • M-PESTI Protocol Phase section
	  • Discovery Phase section
	  • Active Phase section
	  • Target Reset section
	• Removed the Multiple Targets section and added the Secondary Wire Capability with Stimulus and Response section.
	• Updated the following figures in the M-PESTI Protocol section:
	  • Figure 2.2 M-PESTI UART Frame
	  • Figure 2.3 M-PESTI Protocol Timing Diagram
	  • Figure 2.4 Single Initiator Multiple Targets Transaction
	  • Figure 2.5 Manual Multiple Virtual Bytes Program Flow
	  • Figure 2.6 Broadcast Command
	  • Figure 2.7 Abort Mechanism
	  • Figure 2.8 User Command Program Flow
	• Added Figure 2.9 Secondary Wire Capability Program Flow

IP Parameter Description	Updated Table 3.1. General Attributes.
Signal Description	Added the following ports to Table 4.1. Ports Description: • s_mpesti_io • s_mpesti_i • s_mpesti_oe • s_mpesti_o
Register Description	• Updated the introductory paragraph and the examples. • Updated Table 5.1. Register Address Map. • Updated the following sections:

Section	Change Summary
	• Configuration 0x004 section. • Software Reset 0x008 section • Interrupt Status 0x018 section • Interrupt Enable 0x01C section • Interrupt Set 0x020 section • Secondary Wire Control Status 0x040 section • Target Select 0x100 section • User Command 0x104 section • M-PESTI Target 0 Status 0x400+(N*16) section • M-PESTI Target 0 Control Status 0x404+(N*16) section • M-PESTI Target 0 Virtual Wire Input 0x800+(N*16) section • M-PESTI Target 0 Virtual Wire Output 0xC00+(N*16) section • Added the following sections: • Secondary Wire Select 0x044 section • Target Select 0x100 section • User Command 0x104 section • User Write Data 0x108 section • User Read Data 0x10C section
Example Design	• Added a brief introduction to this section and list all the evaluation boards used to run the example design. • Added the following parameters in Table 6.1. M-PESTI Initiator IP Configuration Supported by the Example Design: • Total Number of Secondary Wires • Enable CSR for Secondary Wire • Allow Multiple User Receive Bytes • Maximum User Receive Byte Size • Updated the Maximum Static Discovery Payload Size in Table 6.1. M-PESTI Initiator IP Configuration Supported by the Example Design. • Updated the steps to generate example design and the following figures: • Figure 6.4 Select Template • Figure 6.5 Create SoC Project • Figure 6.6 Define Instance
Designing with the IP	Updated the following figures: • Figure 7.1. Module/IP Block Wizard • Figure 7.2. M-PESTI Initiator IP Core Configuration Example • Figure 7.3. Check Generated Result
Appendix A. Resource Utilization	Updated Table A.1. Resource Utilization for LFMXO5-25-7BBG400I Device.
References	• Added the following references: • Avant-E web page • Avant-G web page • MachXO5-NX web page • MachXO3D web page • MachXO3 web page • Mach-NX web page • Lattice Solutions Reference Designs web page • Lattice Solutions Boards web page • Lattice Solutions Demonstrations web page • Removed the reference to the Lattice Radiant Software User Guide.

Revision 1.0, IP v1.0.0, June 2024

Section	Change Summary

All	Initial release.

www.latticesemi.com