RISC-V SM CPU IP – Lattice Propel Builder 2025.1

IP Version: v1.8.0

User Guide

FPGA-IPUG-02279-1.0

June 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

Contents 3
Abbreviations in This Document 5
1. Introduction 6
1.1. Quick Facts 6
1.2. Features 6
1.3. Conventions 6
1.3.1. Nomenclature 6
1.4. Licensing and Ordering Information 7
2. Functional Descriptions 8
2.1. Overview 8
2.2. Modules Description 9
2.2.1. RISC-V SM CPU Core 9
2.2.2. Submodule (PIC/Timer) 11
2.3. Signal Description 14
2.3.1. Clock and Reset 14
2.3.2. Instruction and Data Interface 14
2.3.3. Interrupt Interface 15
2.4. Attribute Summary 15
3. RISC-V SM CPU IP Generation 17
Appendix A. Resource Utilization 20
Appendix B. Debug with Soft JTAG 21
References 23
Technical Support Assistance 24
Revision History 25

Figures

Figure 2.1. RISC-V SM CPU IP Diagram 8
Figure 2.2. RISC-V SM CPU Core Block Diagram 9
Figure 2.3. PIC Block Diagram 11
Figure 2.4. Timer Block Diagram 13
Figure 3.1. Entering Component Name 17
Figure 3.2. Configuring Parameters 17
Figure 3.3. Verifying Results 18
Figure 3.4. Specifying Instance Name 18
Figure 3.5. Generated Instance 19
Figure B.1. Exporting Pins 21
Figure B.2. Assigning Pins 21
Figure B.3. Setting Environment Variables 22

Tables

Table 1.1 RISC-V SM CPU IP Core Quick Facts 6
Table 2.1. RISC-V SM CPU Core Control and Status Registers 10
Table 2.2. PIC Registers 12
Table 2.3. Timer Registers 14
Table 2.4. Clock and Reset Ports 14
Table 2.5. Instruction Ports 14
Table 2.6. Data Ports 15
Table 2.7. Interrupt Ports 15
Table 2.8. Configurable Attributes 15
Table 2.9. Attributes Description 16
Table A.1. Resource Utilization in MachXO2 Device 20
Table A.2. Resource Utilization in MachXO3D Device 20
Table A.3. Resource Utilization in CrossLink-NX Device 20
Table A.4. Resource Utilization in CertusPro-NX Device 20
Table A.5. Resource Utilization in Lattice Avant Device 20

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
AHB-L	Advanced High-performance Bus – Lite
CPU	Central Processing Unit
CSR	Control and Status Register
DMIPS	Dhrystone MIPS
FPGA	Field Programmable Gate Array
GDB	Gnu Debugger
HDL	Hardware Description Language
IP	Intellectual Property
IRQ	Interrupt Request
ISA	Instruction Set Architecture
JTAG	Joint Test Action Group
LUT	Lookup-Table
MIPS	Million Instructions per Second
SM	State Machine, RISC-V for state machine applications
OpenOCD	Open On-Chip Debugger
PIC	Programmable Interrupt Controller
RISC-V	Reduced Instruction Set Computer-V (Five)
RV32I	RISC-V Integer Instruction Sets
WFI	Wait for Interrupt

1. Introduction

The Lattice Semiconductor RISC-V SM CPU IP contains a 32-bit RISC-V processor core and optional submodules – Timer and Programmable Interrupt Controller (PIC). The CPU core supports the RV32I instruction set, external interrupt, and debug feature, which is JTAG – IEEE 1149.1 compliant. The Timer submodule is a 64-bit real-time counter, which compares a real-time register with another register to assert the timer interrupt. The PIC submodule aggregates up to eight external interrupt inputs into one external interrupt. The submodule registers are accessed by the processor core using a 32-bit Advanced High-performance Bus – Lite (AHB-L) interface.

The design is implemented using Verilog HDL, and it can be configured and generated using the Lattice Propel™ Builder software. It supports Certus™-N2, iCE40 UltraPlus™, Lattice Avant™, MachXO5™-NX, CrossLinkU™-NX, CrossLink™-NX, CertusPro™-NX, Certus-NX, MachXO3D™, MachXO3™, and MachXO2™ FPGA family devices.

1.1. What's New in This IP Release

- Changed the attribute name *Number of Interrupt Requests* to *Number of PIC Interrupt Requests Input* and updated its minimal value from *2* to *0*.
- Added the *Reset Vector* attribute for setting the value of the reset vector.

1.2. Quick Facts

Table 1.1 presents a summary of the RISC-V SM CPU IP.

Table 1.1 RISC-V SM CPU IP Quick Facts

IP Requirements	Supported Devices	Certus-N2, iCE40 UltraPlus, Lattice Avant, MachXO5-NX, CrossLinkU-NX, CrossLink-NX, CertusPro-NX, Certus-NX, MachXO3D, MachXO3L™, MachXO3LF™, MachXO2
Resource Utilization	Supported User Interfaces	AHB-L Interface
	Resources	See Table A.1, Table A.2, Table A.3, Table A.4, and Table A.5
Design Tool Support	Lattice Implementation	IP v1.8.0 – Lattice Propel Builder 2025.1, Lattice Radiant™ Software 2025.1
	Simulation	For a list of supported simulators, see the Lattice Radiant and Lattice Diamond™ software user guide.

1.3. Features

The RISC-V SM CPU IP has the following features:

- RV32I instruction set
- Five stage pipeline
- Support the AHB-L bus standard for instruction or data ports
- Optional debug through Gnu Debugger (GDB) and Open On-Chip Debugger (OpenOCD)
- Optional Timer module
- Optional PIC module
- Interrupt and exception handling with Machine mode in RISC-V privileged ISA Specification Revision 1.10
- About 0.5 DMIPS/MHz performance
- Around 40 MHz in MachXO2 devices, 50 MHz in MachXO3D devices, and 100 MHz in CrossLink-NX devices, tested on Hello World templates provided by Lattice Propel design environment.
- Configurable reset vector

1.4. Conventions

1.4.1. Nomenclature

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

1.4.2. Signal Names

- _n are active low, asserted when value is logic 0.
- _i are input signals.
- _o are output signals.
- _io are bidirectional signals.

1.5. Licensing and Ordering Information

The SM CPU IP is provided at no additional cost with the Lattice Propel design environment. The IP can be fully evaluated in hardware without requiring an IP license string.

2. Functional Descriptions

2.1. Overview

The RISC-V SM CPU IP processes data and instructions while monitoring the external interrupts. As shown in Figure 2.1, the CPU IP has a 32-bit processor core and optional submodules. It uses one read-only AHB-L interface for instruction fetch and another AHB-L interface with Read/Write access for data access. See Table 2.5 and Table 2.6 for the AHB-L Instruction Fetch and Data Accessing ports definition. The CPU core, PIC, Timer, and AHB-L multiplexor run in the system clock domain. The Core Debug runs in both the system clock domain and the JTAG clock domain.

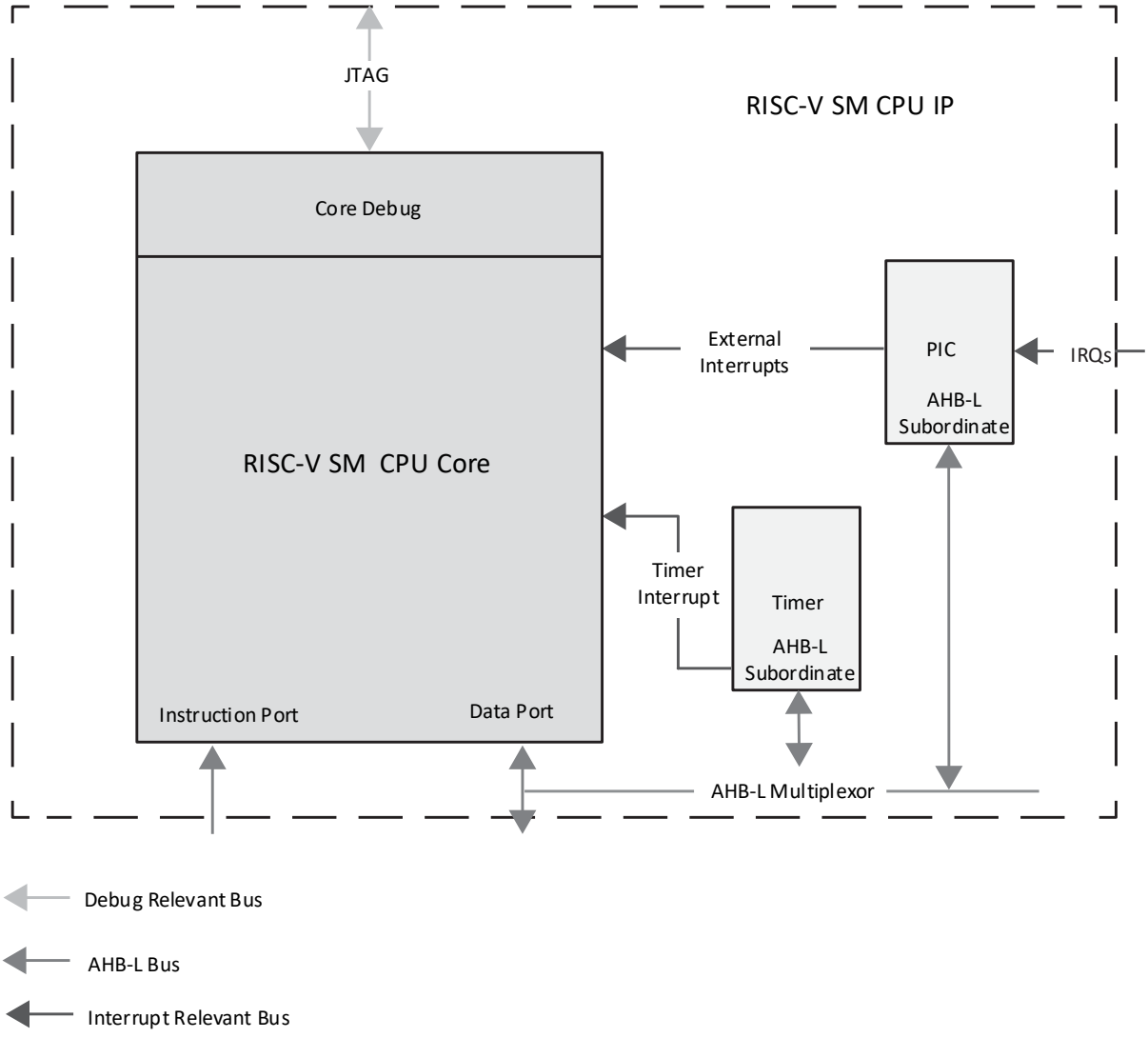


Figure 2.1. RISC-V SM CPU IP Diagram

2.2. Modules Description

2.2.1. RISC-V SM CPU Core

The processor core follows the RV32I instruction set. Figure 2.2 shows the processor core block diagram.

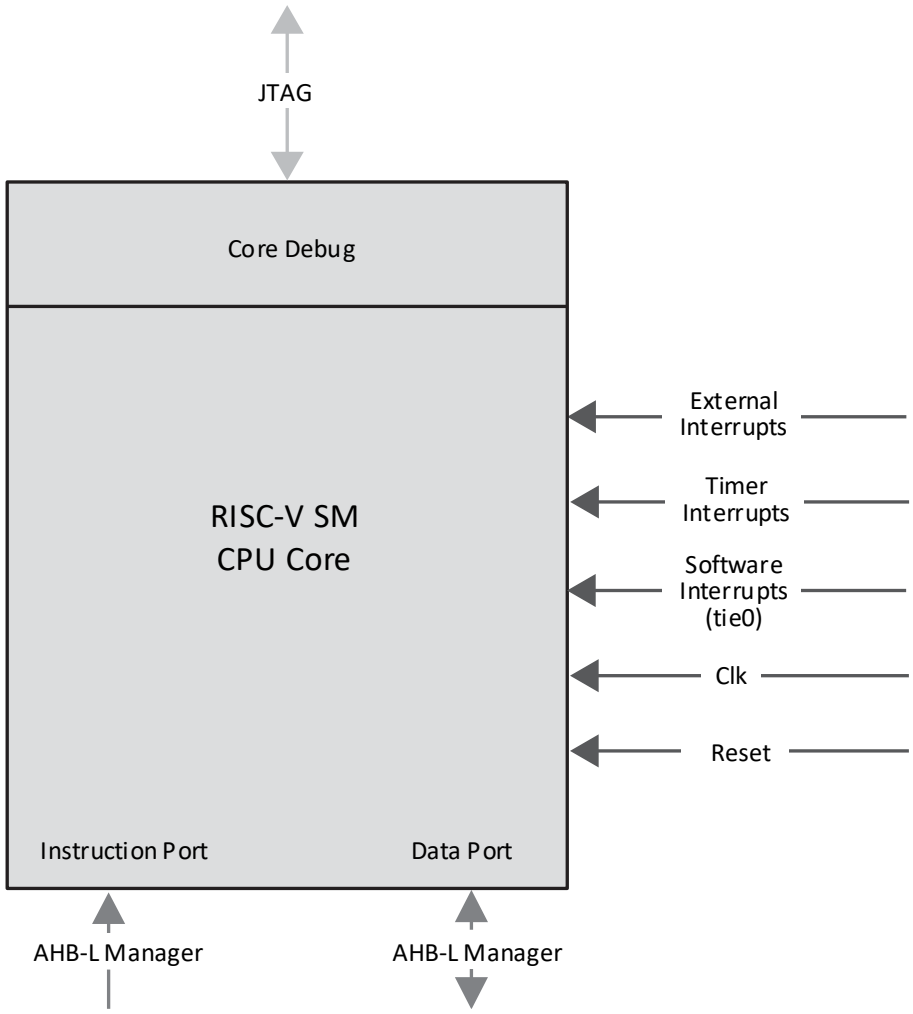


Figure 2.2. RISC-V SM CPU Core Block Diagram

2.2.1.1. Interrupt

The core CPU's interrupts are level sensitive and high active. A given interrupt should remain asserted until cleared by the corresponding interrupt service routine.

If an interrupt occurs, before jumping to the interrupt service routine, the processor core stops the prefetch stage and waits for all instructions in the later pipeline stages to complete their execution.

2.2.1.2. Exception

If an exception occurs, the processor core stops the corresponding instruction, flushes all previous instructions, and waits until the terminated instruction reaches the writeback stage before jumping to the exception service routine.

2.2.1.3. Low Power Mode

The processor core enters into the low power mode with Wait For Interrupt (WFI) instruction. The program counter halts during low power mode, and the processor wakes up if there is an external or timer interrupt.

2.2.1.4. Debug

The processor core supports the IEEE-1149.1 JTAG debug logic with two hardware breakpoints and unlimited software breakpoint. To use the debug module, it is required to allow writes from the data port to the instruction memory in the SoC. Single-port instruction memory is not allowed to debug. For implementing software breakpoint, the processor needs a data path to swap the target instruction with the EBREAK instruction in the instruction memory.

The soft JTAG is available on Certus-N2, Lattice Avant, Certus-NX, CertusPro-NX, CrossLink-NX, and MachXO5-NX devices. For more information, refer to Appendix B.

2.2.1.5. Reset Vector

The reset vector of the processor is configurable through the Reset Vector attribute in the Module/IP Block Wizard GUI.

2.2.1.6. Control and Status Registers

The processor core supports the Control and Status Registers (CSRs) listed in Table 2.1.

Table 2.1. RISC-V SM CPU Core Control and Status Registers

CSR Number	CSR Name	Access	Fields
0x300	Machine Status (mstatus)	read/write	bits[12:11]: mpp, privilege mode before entering a trap. It should always be 2'b11 in machine mode in this CPU core. bit [7]: mpie, mie before entering a trap. Updated to mie value when entering a trap. bit [3]: mie, global interrupt enable.
0x301	Machine ISA (misa)	read-only	bit[31:30]: base, hardwired 0x1, which stands for RV32. bit[25:0]: extension, hardwired 0x100, which stands for RV32I.
0x304	Machine Interrupt Enable (mie)	read/write	bit[11]: meie, machine mode external interrupt enable. bit[7]: mtie, machine mode timer interrupt enable. bit[3]: msie, machine mode software interrupt enable.
0x305	Machine Trap-Vector Base-Address (mtvec)	cannot be accessed, fixed to 0x20	bit[31:2]: trap vector base address, 4-byte aligned. bit[1:0]: trap vector mode. All traps set the program counter to the base address in RISC-V SM CPU core.
0x341	Machine Exception Program Counter (mepc)	cannot be accessed	bits[31:0]: when trap is taken into machine mode, mepc is used to store the address of the instruction that encounters the exception.
0x342	Machine Cause (mcause)	read-only	bit[31]: 1'b1 – interrupt, 1'b0 – exception bit[3:0]: exception code For interrupt: 3 – Machine software interrupt 7 – Machine timer interrupt 11 – Machine external interrupt For exception: 0 – Instruction address misaligned 1 – Instruction access fault 2 – Illegal instruction 4 – Load address misaligned 5 – Load access fault
0x343	Machine Trap Value (mtval)	cannot be accessed	mtval stores the bad address when an exception occurs in machine mode. The source of bad address can be: • exceptions on dbus, such as access error, mmu exception, or unaligned access. In this condition, it contains dbus address. • exceptions from CSRs, such as ecall and ebreak. In this condition, it contains the execute instruction. • exceptions by fetch. It contains fetch pc. • exceptions by decode. It contains the decoded instruction itself, not the address of the instruction.

CSR Number	CSR Name	Access	Fields
0x344	Machine Interrupt Pending (mip)	read/write	bit[11]: meip, machine mode external interrupt pending, read-only. bit[7] mtip, machine mode timer interrupt pending, read-only. bit[3] msip, machine mode software interrupt pending, readable and writable.

2.2.1.7. System Reset Output

The system_resetn_o signal is driven by two ways. When debug is not enabled or if debug reset is not issued, system_resetn_o is the passed value from the input reset signal rst_n_i. It is asynchronous with input clock. When the debugger is enabled and debug reset is issued, the debug reset signal is synchronized to system clock domain and the system_resetn_o is the output of the synchronized signal.

2.2.2. Submodule

The CPU soft IP contains submodules: PIC and Timer. The PIC and Timer share the same start address in memory map and a fixed 2 KB address range is allocated, if either PIC or Timer is enabled.

2.2.2.1. PIC

The PIC aggregates up to eight external interrupt inputs (IRQs) into one interrupt output to the processor core. The PIC's interrupt status register can be used to read the values of IRQs. Individual IRQs can be configured by programming the corresponding PIC_STATUS, PIC_ENABLE, PIC_SET, and PIC_POL registers. All registers can be accessed through the CPU's internal AHB-L interface, as shown in Figure 2.3.

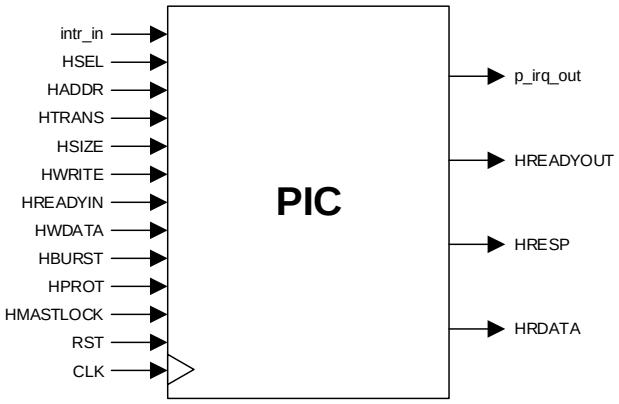


Figure 2.3. PIC Block Diagram

Table 2.2 provides the descriptions of PIC registers.

Table 2.2. PIC Registers

<table>
<tr><th>Offset</th><th>Name</th><th>Description</th></tr>
<tr><td>0x000</td><td>PIC_STATUS</td><td>Interrupt Status Register
Access: read-write
Parameterizable width: min=2, max=16
Indicates the pending interrupt at the corresponding interrupt request port, irq[i] at top level.

<table>
<tr><th>Field</th><th>Name</th><th>Access</th><th>Width</th><th>Reset</th></tr>
<tr><td>[N-1]</td><td>PIC_STATUS [N-1]</td><td>RW</td><td>1</td><td>0x0</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td></tr>
<tr><td>[1]</td><td>PIC_STATUS [1]</td><td>RW</td><td>1</td><td>0x0</td></tr>
<tr><td>[0]</td><td>PIC_STATUS [0]</td><td>RW</td><td>1</td><td>0x0</td></tr>
</table>

PIC_STATUS[i]:
Read
• 0 – No interrupt at irq[i].
• 1 – Interrupt pending at irq[i].
Write
• 0 – No effect.
• 1 – Clear interrupt status for irq[i].</td></tr>
<tr><td>0x004</td><td>PIC_ENABLE</td><td>Interrupt Enable Register
Access: read-write
Parameterizable width: min=2, max=16
Indicates whether the processor responds to the interrupt from the corresponding interrupt request port irq[i] or not.

<table>
<tr><th>Field</th><th>Name</th><th>Access</th><th>Width</th><th>Reset</th></tr>
<tr><td>[N-1]</td><td>PIC_ENABLE[N-1]</td><td>RW</td><td>1</td><td>0x0</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td></tr>
<tr><td>[1]</td><td>PIC_ENABLE[1]</td><td>RW</td><td>1</td><td>0x0</td></tr>
<tr><td>[0]</td><td>PIC_ENABLE[0]</td><td>RW</td><td>1</td><td>0x0</td></tr>
</table>

PIC_ENABLE[i]:
Read
• 0 – irq[i] disabled.
• 1 – irq[i] enabled.
Write
• 0 – Disable irq[i].
• 1 – Enable irq[i].</td></tr>
<tr><td>0x008</td><td>PIC_SET</td><td>Interrupt Set Register
Access: write-only
Parameterizable width: min=2, max=16
Sets the interrupt status for the corresponding interrupt request port irq[i].

<table>
<tr><th>Field</th><th>Name</th><th>Access</th><th>Width</th><th>Reset</th></tr>
<tr><td>[N-1]</td><td>PIC_SET [N-1]</td><td>W</td><td>1</td><td>0x0</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td></tr>
<tr><td>[1]</td><td>PIC_SET [1]</td><td>W</td><td>1</td><td>0x0</td></tr>
<tr><td>[0]</td><td>PIC_SET [0]</td><td>W</td><td>1</td><td>0x0</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Offset</th><th>Name</th><th>Description</th></tr>
<tr><td></td><td></td><td>PIC_SET[i]:
Read
• Invalid operation. Can get 0.
Write
• 0 – No effect.
• 1 – Set interrupt status for irq[i], that is, set PIC_STATUS[i].</td></tr>
<tr><td>0x00C</td><td>PIC_POL</td><td>Interrupt Polarity Register
Access: read-write
Parameterizable width: min=2, max=16
Indicates the polarity of corresponding interrupt request port, irq[i].

<table>
<tr><th>Field</th><th>Name</th><th>Access</th><th>Width</th><th>Reset</th></tr>
<tr><td>[N]</td><td>PIC_POL [N]</td><td>RW</td><td>1</td><td>0x0</td></tr>
<tr><td>…</td><td>…</td><td>…</td><td>…</td><td>…</td></tr>
<tr><td>[1]</td><td>PIC_POL l[1]</td><td>RW</td><td>1</td><td>0x0</td></tr>
<tr><td>[0]</td><td>PIC_POL [0]</td><td>RW</td><td>1</td><td>0x0</td></tr>
</table>

PIC_POL[i]:
Read
• 0 – irq[i] is active high.
• 1 – irq[i] is active low.
Write
• 0 – Set irq[i] active high.
• 1 – Set irq[i] active low.</td></tr>
</table>

Note: The register definition of PIC follows Lattice Interrupt Interface (LINTR) Standard, refer to Lattice Memory Mapped Interface and Lattice Interrupt Interface User Guide (FPGA-UG-02039) for more information.

2.2.2.2. Timer

The Timer module provides a 64-bit real-time counter register (mtime) and time compare register (mtimecmp). An output interrupt signal notifies the RISC-V processor core when the value of mtime is greater than or equal to the value of mtimecmp. All registers can be accessed through the CPU's internal AHB-L interface, as shown in Figure 2.4.

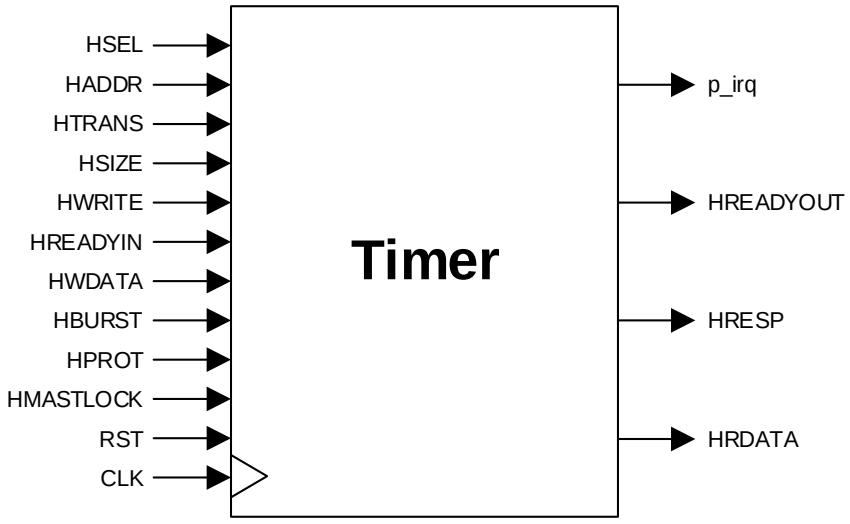


Figure 2.4. Timer Block Diagram

Table 2.3 provides the descriptions of Timer registers.

Table 2.3. Timer Registers

Offset	Name	Description
0x400	TIMER_CNT_L	Lower 32 bits of the Timer module counter register. Field | Name | Access | Width | Reset [63:0] | mtime | RW | 64 | 0x0 mtime A 64-bit real-time counter register. You must set the register to a non-zero value to start the counting process.
0x404	TIMER_CNT_H	Higher 32 bits of the Timer module counter register.
0x410	TIMER_CMP_L	Lower 32 bits for the Timer module time compare register. Field | Name | Access | Width | Reset [63:0] | mtimecmp | RW | 64 | 0x0 mtimecmp This register is used to generate or clear the timer interrupt (mtip). When the value of the mtime register is greater than or equal to the value of the mtimecmp register, TIMER_IRQ_M0 is asserted. The interrupt remains posted until mtimecmp becomes greater than mtime, typically as a result of writing mtimecmp.
0x414	TIMER_CMP_H	Higher 32 bits for the Timer module time compare register.

2.3. Signal Description

Table 2.4 to Table 2.7 list the ports of the CPU soft IP in different categories.

2.3.1. Clock and Reset

Table 2.4. Clock and Reset Ports

Name	Direction	Width	Description
clk_i	In	1	RISC-V soft IP clock.
rst_n_i	In	1	Global reset, active low.
system_resetn_o	Out	1	Combined Global reset and Debug Reset from JTAG, active low.

2.3.2. Instruction and Data Interface

Table 2.5. Instruction Ports

Name	Direction	Width	Description
AHBL_M0_INSTR - HADDR	Out	32	—
AHBL_M0_INSTR - HWRITE	Out	1	Fixed to 1’b0
AHBL_M0_INSTR - HSIZE	Out	3	Fixed to 3’b010
AHBL_M0_INSTR - HPROT	Out	4	Fixed to 4’b1110
AHBL_M0_INSTR - HTRANS	Out	2	—
AHBL_M0_INSTR - HBURST	Out	3	Fixed to 3’b000
AHBL_M0_INSTR - HMASTLOCK	Out	1	Fixed to 1’b0
AHBL_M0_INSTR - HWDATA	Out	32	—

Name	Direction	Width	Description
AHBL_M0_INSTR - HRDATA	In	32	—
AHBL_M0_INSTR - HREADY	In	1	—
AHBL_M0_INSTR - HRESP	In	1	—

Table 2.6. Data Ports

Name	Direction	Width	Description
AHBL_M1_DATA - HADDR	Out	32	—
AHBL_M1_DATA - HWRITE	Out	1	—
AHBL_M1_DATA - HSIZE	Out	3	—
AHBL_M1_DATA - HPROT	Out	4	Fixed to 4'b1111
AHBL_M1_DATA - HTRANS	Out	2	—
AHBL_M1_DATA - HBURST	Out	3	Fixed to 3'b000
AHBL_M1_DATA - HMASTLOCK	Out	1	—
AHBL_M1_DATA - HSEL	Out	1	—
AHBL_M1_DATA - HWDATA	Out	32	—
AHBL_M1_DATA - HRDATA	In	32	—
AHBL_M1_DATA - HREADY	In	1	—
AHBL_M1_DATA - HRESP	In	1	—

For more information, refer to AMBA 3 AHB-Lite Protocol V1.0.

2.3.3. Interrupt Interface

Table 2.7. Interrupt Ports

Name	Type	Width	Description
IRQ_S_{0-15}	In	1	Peripheral interrupts.
TIMER_IRQ_M0	Out	1	Timer interrupt output, active high. It exists only when Enable Timer is checked.
TIMER_IRQ_S0	In	1	Timer interrupt input, active high. It exists only when Enable Timer is unchecked.

2.4. Attribute Summary

The configurable attributes of the RISC-V SM CPU IP are shown in Table 2.8 and are described in Table 2.9.

The attributes can be configured through the Lattice Propel Builder software.

Table 2.8. Configurable Attributes

Attribute	Selectable Values	Default	Dependency on Other Attributes
Reset			
Reset Vector	32'h00000000–32'hFFFFFFFF	32'h00000000	—
General			
Debug Enable	Enabled, Disabled	Enabled	—
Soft JTAG	Enabled, Disabled	Disabled	—
JTAG Channel Selection for Certain Devices	14–16	14	Enabled when Debug Enable is enabled.
PIC Enable	Enabled, Disabled	Enabled	—

Attribute	Selectable Values	Default	Dependency on Other Attributes
Number of PIC Interrupt Requests Input	0–16	8	The value of this attribute is configurable from 1–16 when PIC Enable is enabled. When PIC Enable is disabled, the value is fixed at 0.
Timer Enable	Enabled, Disabled	Enabled	—
PIC and Timer Base Address	32'h00000000–32'hFFFFF800	32h'FFFF0000	Enabled when PIC Enable or Timer Enable is enabled.

Table 2.9. Attributes Description

Attribute	Description
Reset	
Reset Vector	The value of the reset vector
General	
Debug Enable	1: The Debug function is enabled. 0: The Debug function is disabled.
Soft JTAG	1: Debug with hard JTAG. 0: Debug with soft JTAG.
JTAG Channel Selection for Certain Devices	Specifies the channel of SM JTAG block.
PIC Enable	1: PIC is enabled. 0: PIC is disabled.
Number of PIC Interrupt Requests Input	The number of peripheral interrupts
Timer Enable	1: Timer is enabled. 0: Timer is disabled.
PIC and Timer Base Address	Start address of PIC and Timer

3. RISC-V SM CPU IP Generation

This section provides information on how to generate the CPU IP module using Lattice Propel Builder software.

To generate the CPU IP module:

1. In Lattice Propel Builder software, create a new design. Select the CPU package.
2. Enter the component name. Click **Next**, as shown in Figure 3.1.

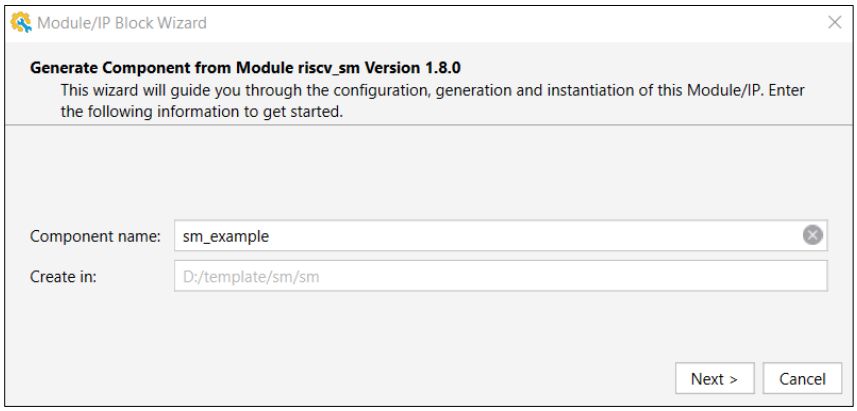


Figure 3.1. Entering Component Name

3. Configure the parameters as needed. Click **Generate** (Figure 3.2).

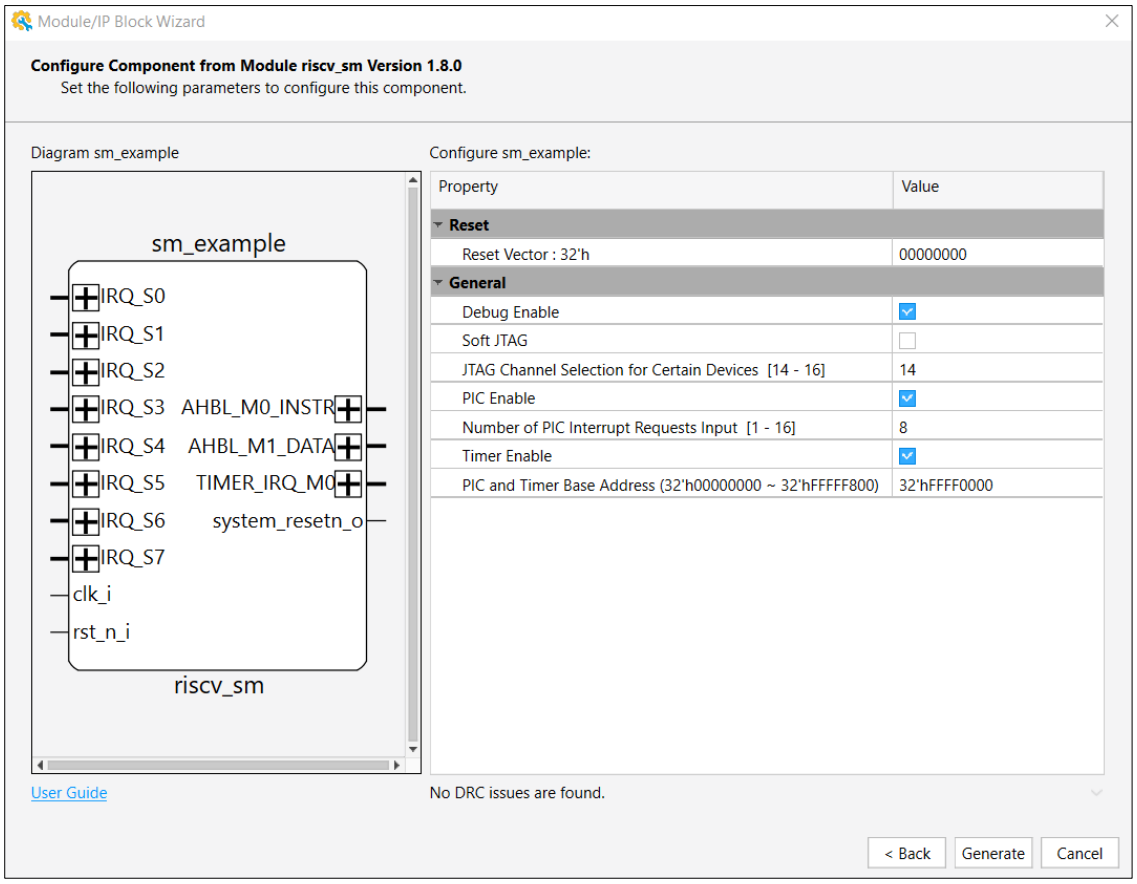


Figure 3.2. Configuring Parameters

4. Verify the information. Click **Finish** (Figure 3.3).

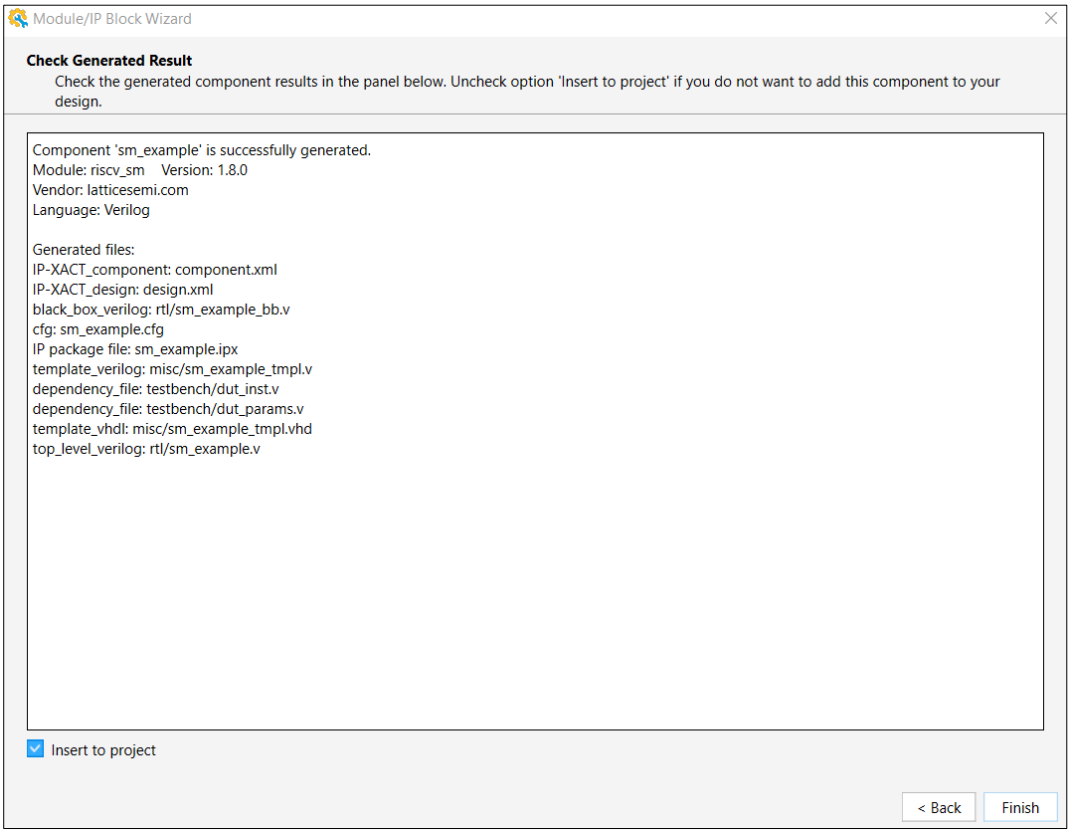


Figure 3.3. Verifying Results

5. Confirm or modify the module instance name. Click **OK** (Figure 3.4).

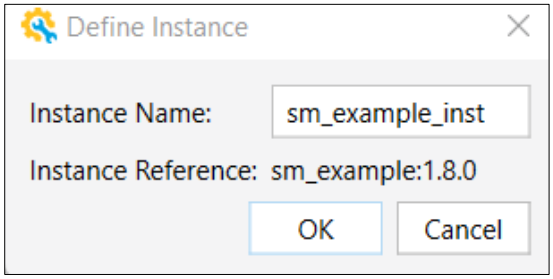


Figure 3.4. Specifying Instance Name

6. The CPU IP instance is successfully generated, as shown in Figure 3.5.

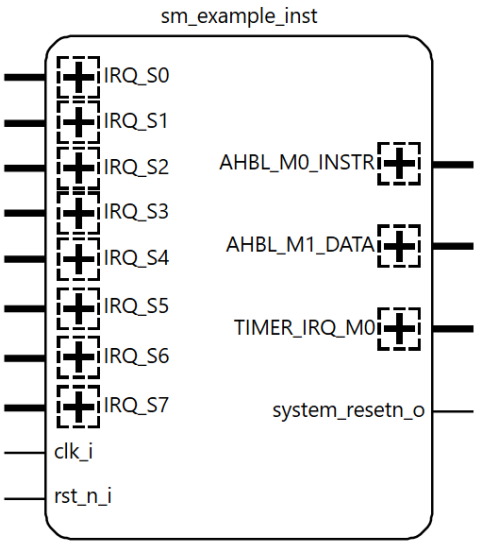


Figure 3.5. Generated Instance

Appendix A. Resource Utilization

Table A.1. Resource Utilization in MachXO2 Device

Configuration	LUTs	Registers	sysMEM EBRs
Processor core only	728	558	4
Processor core + PIC	807	577	4
Processor core + Timer	996	704	4
Processor core + Debug	947	861	4
Processor core + PIC + Timer	1047	713	4
Processor core + PIC + Timer + Debug	1257	1017	4

Note: Resource utilization characteristics are generated using Lattice Diamond software.

Table A.2. Resource Utilization in MachXO3D Device

Configuration	LUTs	Registers	sysMEM EBRs
Processor core only	760	568	4
Processor core + PIC	850	587	4
Processor core + Timer	1135	714	4
Processor core + Debug	1012	871	4
Processor core + PIC + Timer	1186	725	4
Processor core + PIC + Timer + Debug	1402	1026	4

Note: Resource utilization characteristics are generated using Lattice Diamond software.

Table A.3. Resource Utilization in CrossLink-NX Device

Configuration	LUTs	Registers	sysMEM EBRs
Processor core only	899	596	2
Processor core + PIC	980	615	2
Processor core + Timer	1365	715	2
Processor core + Debug	1233	974	2
Processor core + PIC + Timer	1382	724	2
Processor core + PIC + Timer + Debug	1721	1127	2

Note: Resource utilization characteristics are generated using Lattice Radiant software.

Table A.4. Resource Utilization in CertusPro-NX Device

Configuration	LUTs	Registers	sysMEM EBRs
Processor core only	996	570	2
Processor core + PIC	1129	607	2
Processor core + Timer	1369	726	2
Processor core + Debug	1257	947	2
Processor core + PIC + Timer	1466	742	2
Processor core + PIC + Timer + Debug	1652	1119	2

Note: Resource utilization characteristics are generated using Lattice Radiant software.

Table A.5. Resource Utilization in Lattice Avant Device

Configuration	LUTs	Registers	sysMEM EBRs
Processor core only	1045	570	2
Processor core + PIC	1125	607	2
Processor core + Timer	1347	715	2
Processor core + Debug	1352	997	2
Processor core + PIC + Timer	1443	742	2
Processor core + PIC + Timer + Debug	1730	1118	2

Note: Resource utilization characteristics are generated using Lattice Radiant software.

Appendix B. Debug with Soft JTAG

To debug with Soft JTAG:

1. In Lattice Propel Builder software, enable **Soft JTAG** in the IP block Wizard GUI (Figure 3.2) when generating the IP.
2. After the IP is generated, right-click on the **JTAG** port and select **Export** (Figure B.1).

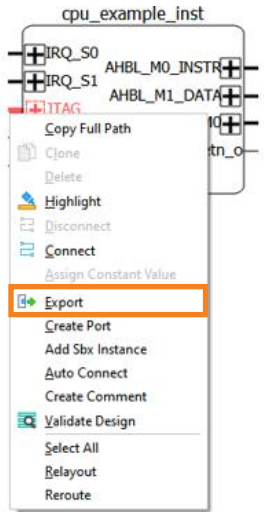
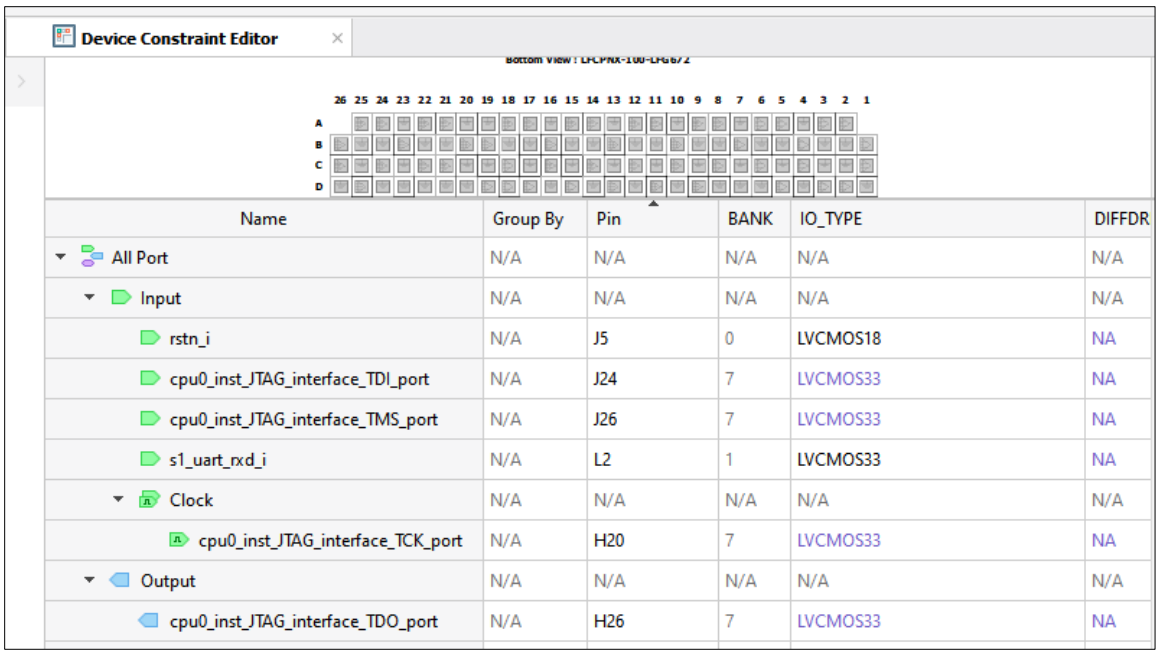


Figure B.1. Exporting Pins

3. Assign the normal I/O as JTAG I/O using the Device Constraint Editor in Lattice Radiant software.
   a. Synthesize the design SoC in Lattice Radiant software by clicking **Synthesis Design** from the process toolbar.
   b. Open **Device Constraint Editor** from the **Tools** tab in Lattice Radiant software and assign the pins. For different devices, refer to the user guide of each board. The following assignment is for LFCPNX-100-9LF672C (Figure B.2).



Name	Group By	Pin	BANK	IO_TYPE	DIFFDR
All Port	N/A	N/A	N/A	N/A	N/A
Input	N/A	N/A	N/A	N/A	N/A
rstn_i	N/A	J5	0	LVCMOS18	NA
cpu0_inst_JTAG_interface_TDI_port	N/A	J24	7	LVCMOS33	NA
cpu0_inst_JTAG_interface_TMS_port	N/A	J26	7	LVCMOS33	NA
s1_uart_rxd_i	N/A	L2	1	LVCMOS33	NA
Clock	N/A	N/A	N/A	N/A	N/A
cpu0_inst_JTAG_interface_TCK_port	N/A	H20	7	LVCMOS33	NA
Output	N/A	N/A	N/A	N/A	N/A
cpu0_inst_JTAG_interface_TDO_port	N/A	H26	7	LVCMOS33	NA

Figure B.2. Assigning Pins

c. Double-click on the targeted strategy in the **File List** view to open the **Strategies** dialog box.

d. In the **Strategies** dialog box, set the environment variable for **Place & Route Design**. Enter `-exp WARNING_ON_PCLKPLC1=1` to the Value of **Command Line Options** if TCK connects to normal I/O (Figure B.3).

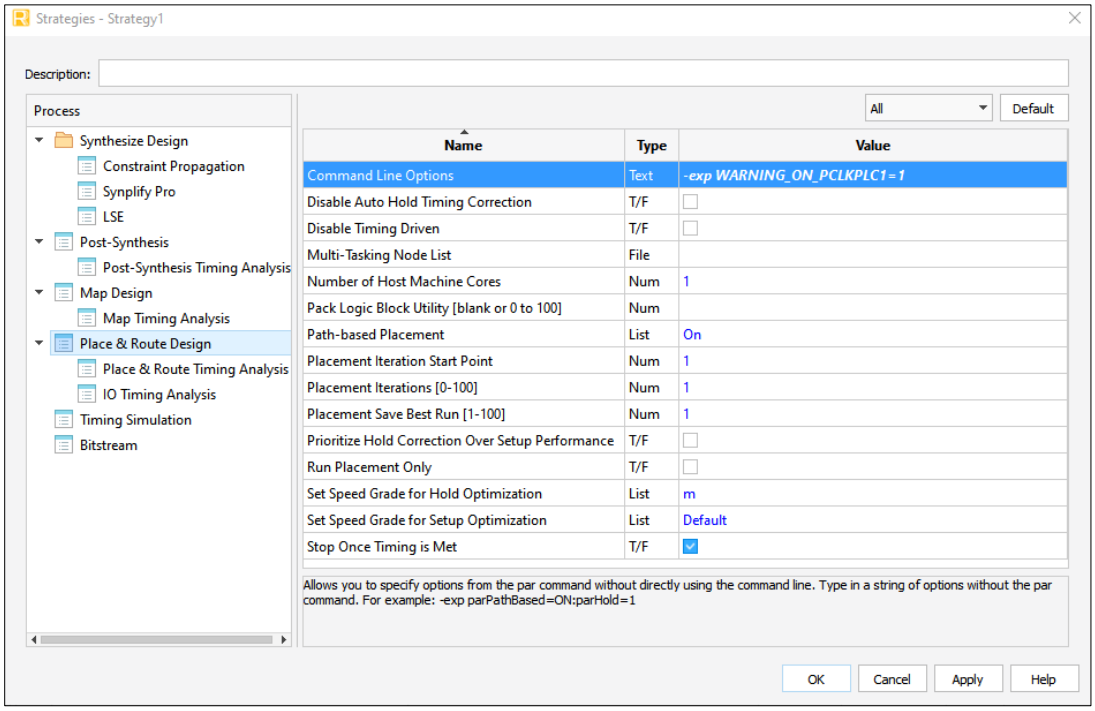


Figure B.3. Setting Environment Variables

e. Generate the bitstream and load it to the board.

f. Connect the pins on cable to the board according to your assignments. Connect VCC and GND. Scan the cable in Propel SDK and ignore the scanning of the device.
Note: C projects generated for Certus-N2 and Lattice Avant family devices cannot use Soft JTAG to debug on MachXO5-NX, Certus-NX, CertusPro-NX, and CrossLink-NX boards and vice versa.

References

- Lattice Propel Builder 2025.1 User Guide (FPGA-UG-02235)
- AMBA 3 AHB-Lite Protocol V1.0
- RISC-V Privileged Specification (20211203)
- RISC-V Instruction Set Manual (20190608)
- Lattice Memory Mapped Interface and Lattice Interrupt Interface User Guide (FPGA-UG-02039)

For more information, refer to:

- Lattice Certus-N2 Family Devices web page
- Lattice Propel web page
- Lattice Avant-E Family Devices web page
- Lattice Avant-G Family Devices web page
- Lattice Avant-X Family Devices web page
- MachXO5-NX Family Devices web page
- Certus-NX Family Devices web page
- CertusPro-NX Family Devices web page
- CrossLink-NX Family Devices web page
- MachXO3D Family Devices web page
- MachXO3 Family Devices web page
- MachXO2 Family Devices web page
- iCE40 UltraPlus Devices web page
- Lattice Insights for Training Series 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 1.0, IP v1.8.0, June 2025

Section	Change Summary
All	Production release.

www.latticesemi.com