RISC-V MC CPU IP Core - Lattice Propel Builder

User Guide

FPGA-IPUG-02149-1.0

November 2020

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 and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. 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. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. 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.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
AHB-L	Advanced High-performance Bus – Lite
CPU	Central Processing Unit
DMIPS	Dhrystone MIPS (Million Instructions per Second)
FPGA	Field Programmable Gate Array
GDB	Gnu Debugger
HDL	Hardware Description Language
IRQ	Interrupt Request
ISA	Instruction Set Architecture
JTAG	Joint Test Action Group
LUT	Lookup-Table
MC	Micro-Controller (RISC-V for Micro-Controller applications)
OpenOCD	Open On-Chip Debugger
PIC	Programmable Interrupt Controller
RISC-V	Reduced instruction set computer-V (five)
RV32IC	RISC-V Integer and Compressed Instruction Sets

1. Introduction

The Lattice Semiconductor RISC-V MC CPU soft IP contains a 32-bit RISC-V processor core and optional submodules – Timer and Programmable Interrupt Controller (PIC). The CPU core supports the RV32IC 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 in Verilog HDL. It can be configured and generated based on Table 1.1.

Table 1.1. FPGA Software for IP Configuration, Generation and Implementation

Supported FPGA Family	IP Configuration and Generation	IP Implementation (Synthesis, Map, Place and Route)
MachXO3D™	Lattice Propel™ Builder software	Lattice Diamond® software
CrossLink™-NX	Lattice Propel™ Builder software	Lattice Radiant™ software
Certus™-NX	Lattice Propel™ Builder software	Lattice Radiant™ software

1.1. Features

The RISC-V MC soft IP has the following features:

- RV32IC instruction set
- Five stages of pipelines
- Support for the AHB-L bus standard for instruction/data port
- Optional debug through GDB and OpenOCD
- Optional Timer/PIC modules
- Interrupt and exception handling with Machine mode in RISC-V privileged ISA Specification Revision 1.10
- > 0.7 DMIPS/MHz performance
- 60 MHz in MachXO3D device, 100 MHz in CrossLink-NX device

1.2. Conventions

1.2.1. Nomenclature

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

2. Functional Descriptions

2.1. Overview

The RISC-V MC soft IP processes data and instruction by considering the timer interrupt and external interrupt. As shown in Figure 2.1, the CPU IP core module has a 32-bit processor core and optional submodules. It uses two interfaces, one AHB-L interface (Read-Only) for instruction and one AHB-L interface (Read/Write Access) for data memory. See Table 2.5 and Table 2.6 for the AHB-L Instruction Fetch and Data Accessing ports definition. RISC-V core, PIC, Timer, and AHB-L multiplex run in the system clock domain. The RISC-V core debug runs in two clock domains: the system clock domain and the JTAG clock domain.

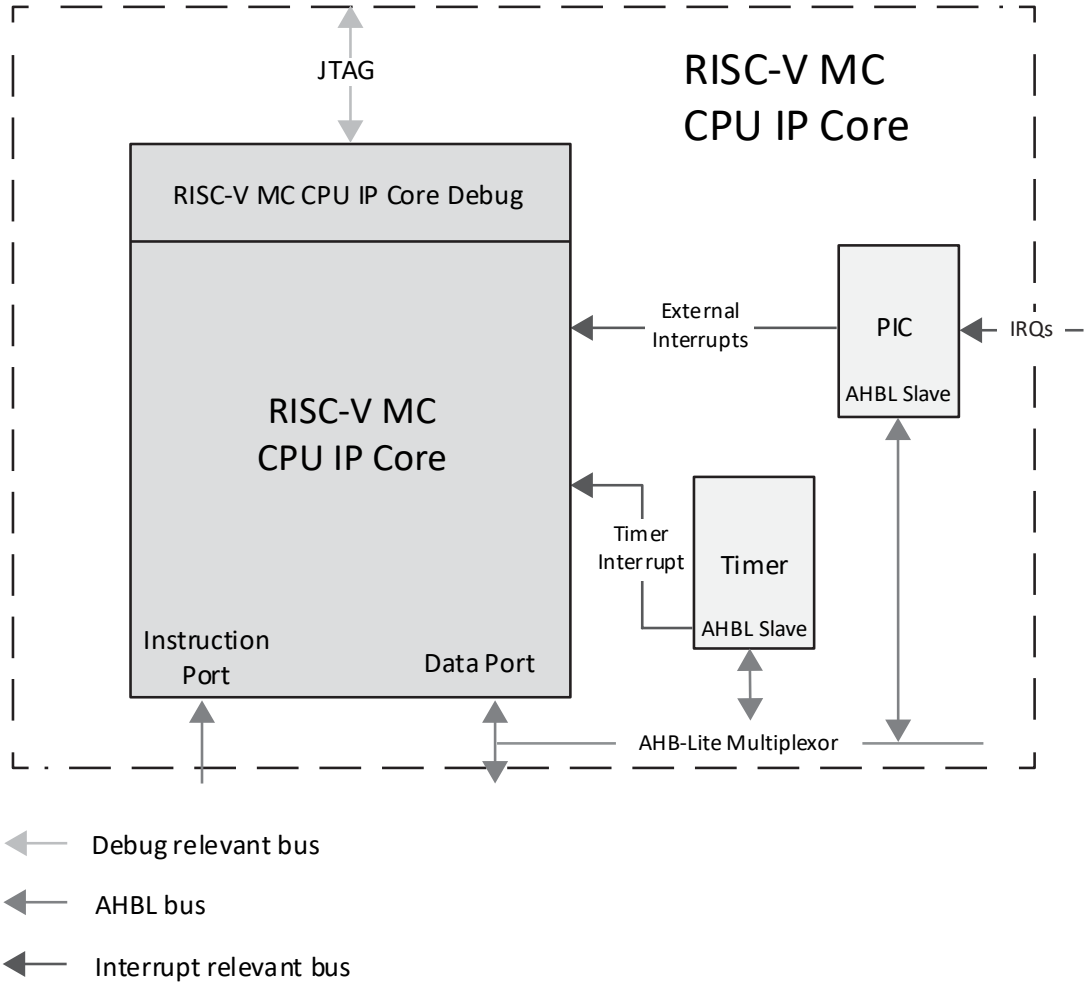

Figure 2.1. RISC-V MC Soft IP Diagram

2.2. Modules Description

2.2.1. RISC-V Processor Core

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

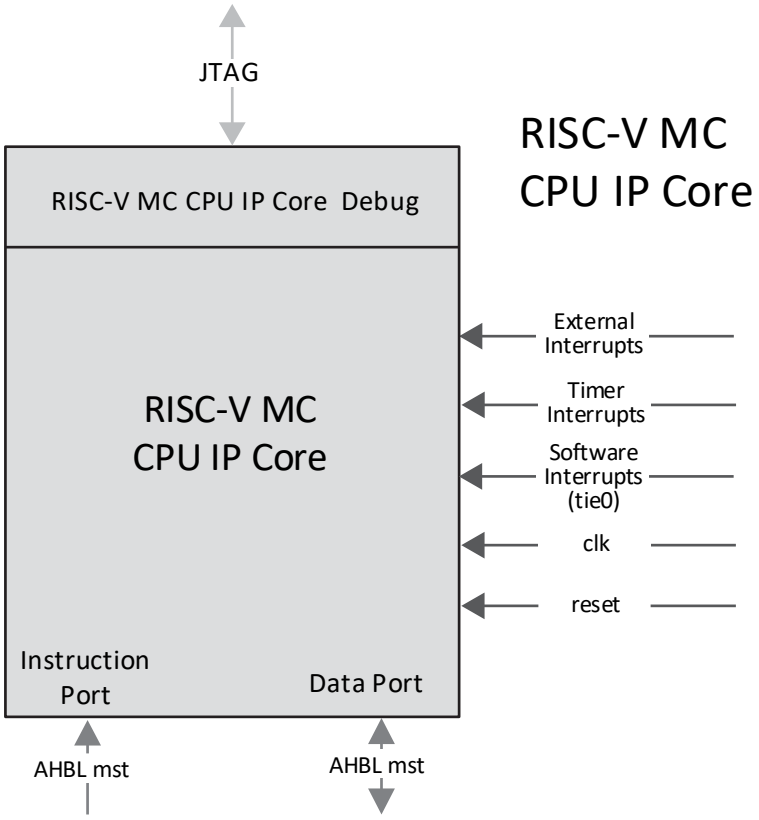

Figure 2.2. RISC-V MC Processor Core Block Diagram

2.2.1.1. Interrupt

Whenever an interrupt occurs, it has to remain in its active level until it is cleared by the processor core interrupt service routine.

If an interrupt occurs before jumping to the interrupt service routine, the processor core stops the pre-fetch 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

Processor core (with *PFR_OPT* unchecked) enters into low power mode with *WFI* command, PC halts during low power mode, processor wakes up if there is external/timer interrupt.

2.2.1.4. Debug

The processor core supports the IEEE-1149.1 JTAG debug logic with two hardware breakpoints.

2.2.1.5. Control and status registers

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

Table 2.1. RISC-V Processor Core Control and Status Registers

addr	CSR Name	Access	Fields
0x300	mstatus (machine status register)	read/write	bits[12:11] mpp: privilege mode before entering trap , should always be 2'b11 (m mode) bit [7] mpie: updated to mie value when entering to trap bit [3] mie: global interrupt enable
0x304	mie (machine interrupt enable register)	read/write	bit[11] meie: m mode external interrupt enable bit[7] mtie: m mode timer interrupt enable bit[3] msie: m mode software interrupt enable
—	mtvec	cannot be accessed (fixed to 0x20)	bit[31:2]: four byte aligned trap vector base address bit[0]: mode trap vector mode: all traps set the program counter to base
0x341	mepc (machine exception program counter)	read only	bits[31:0]: When trap in taken into m mode, mepc is used to store the address of the instruction that encountered exception.
0x342	mcause (machine cause register)	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	mtval (machine trap value register)	read only	bits[31:0]: When a hardware breakpoint is triggered, or an instruction-fetch, load, or store address is misaligned or access exception occurs, mtval is written with the fault address. It may also be written with illegal instruction when an illegal instruction occurs.
0x344	mip (machine interrupt pending register)	read only	bit[11] meip: m mode external interrupt pending bit[7] mtip: m mode timer interrupt pending bit[3] msip: m mode software interrupt pending

2.2.2. Submodule (PIC/Timer)

The CPU soft IP contains submodules: PIC and Timer. The PIC and Timer share the same start address in memory map and a fixed two KB address range is allocated, if any of 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 processor core. The interrupt status register and 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 an AHB-L interface, as shown in Figure 2.3.

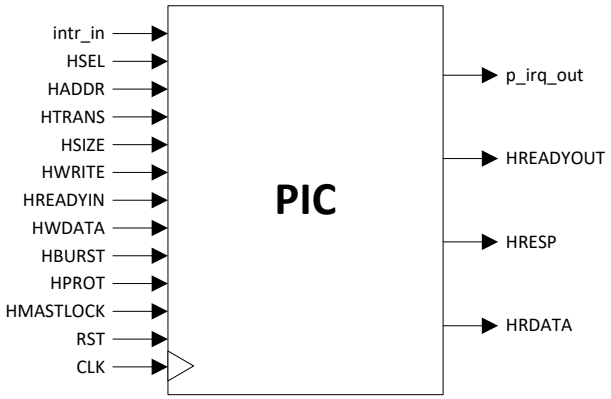

Figure 2.3. PIC Block Diagram

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 (read-write) (parameterizable width, min=2, max=8)
Indicate the pending interrupt at 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_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>
</table>

<table>
<tr><th>Offset</th><th>Name</th><th>Description</th></tr>
<tr><td>0x004</td><td>PIC_ENABLE</td><td>Interrupt Enable Register (read-write) (parameterizable width, min=2, max=8)
Enable or Disable 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_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 (write-only) (parameterizable width, min=2, max=8)
Set the interrupt status for 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>

PIC_SET[i]:
Read
• Invalid operation, gets 0.
Write
• 0 – no effect
• 1 – set interrupt status for irq[i] (set PIC_STATUS[i])</td></tr>
<tr><td>0x00C</td><td>PIC_POL</td><td>Interrupt Polarity Register (read-write) (parameterizable width, min=2, max=8)
Indicates the polarity of corresponding interrupt request (irq[i]) port.

<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 (LMMI) and Lattice Interrupt Interface (LINTR) 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 notices 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 an AHB-L interface, as shown in Figure 2.4.

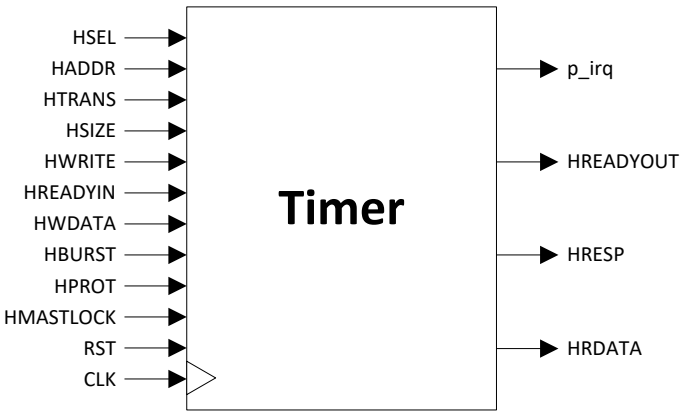

Figure 2.4. Timer Block Diagram

Table 2.3 provides the descriptions of Timer registers.

Table 2.3. Timer Registers

<table>
<tr><th>Offset</th><th>Name</th><th>Description</th></tr>
<tr><td>0x400</td><td>TIMER_CNT_L</td><td>Lower 32 bits of Timer counter register

<table><tr><td>Field</td><td>Name</td><td>Access</td><td>Width</td><td>Reset</td></tr><tr><td>[63:0]</td><td>mtime</td><td>RW</td><td>64</td><td>0x0</td></tr></table>
mtime
A 64-bit real-time counter register. You must set the register to a non-zero value to start the counting process.</td></tr>
<tr><td>0x404</td><td>TIMER_CNT_H</td><td>Higher 32 bits of Timer counter register.</td></tr>
<tr><td>0x410</td><td>TIMER_CMP_L</td><td>Lower 32 bits for Timer time compare register.

<table><tr><td>Field</td><td>Name</td><td>Access</td><td>Width</td><td>Reset</td></tr><tr><td>[63:0]</td><td>mtimecmp</td><td>RW</td><td>64</td><td>0x0</td></tr></table>
mtimecmp
This register is used to generate or clear the timer interrupt (mtip). When the value of mtime register is greater than or equal to the value of mtimecmp register, the cpu_mtip_o is asserted and remains asserted until it is cleared by writing to mtimecmp register. Lower 32 bit for Timer time compare register</td></tr>
<tr><td>0x400</td><td>TIMER_CNT_L</td><td>Higher 32 bits for Timer time compare register</td></tr>
</table>

2.3. Signal Description

Table 2.4 to Table 2.6 list the ports of the CPU soft IP in different categories.

2.3.1. Clock and Reset

Table 2.4. Clock and Reset Port

Name	Type	Width	Description
CLK	In	1	RISC-V soft IP clock
RESETN	In	1	Global reset (active low)
SYSTEM_RESETN	Out	1	Combined Global reset and Debug Reset from JTAG

2.3.2. Instruction and Data Interface

Table 2.5. Instruction Port

Name	Type	Width	Description
INSR_HADDR	Out	32	—
INSR_HWRITE	Out	1	Fixed to 1'b0
INSR_HSIZE	Out	3	Fixed to 3'b010
INSR_HPROT	Out	1	Fixed to 4'b1110
INSR_HTRANS	Out	2	—
INSR_HBURST	Out	3	Fixed to 3'b000
INSR_HMASTLOCK	Out	1	Fixed to 1'b0
INSR_HWDATA	Out	32	—
INSR_HRDATA	In	32	—
INSR_HREADY	In	1	—
INSR_HRESP	In	1	—

Table 2.6. Data Port

Name	Type	Width	Description
DATA_HADDR	Out	32	—
DATA_HWRITE	Out	1	—
DATA_HSIZE	Out	3	—
DATA_HPROT	Out	1	Fixed to 4'b1111
DATA_HTRANS	Out	2	—
DATA_HBURST	Out	3	Fixed to 3'b000
DATA_HMASTLOCK	Out	1	—
DATA_HSEL	Out	1	—
DATA_HWDATA	Out	32	—
DATA_HRDATA	In	32	—
DATA_HREADY	In	1	—

Note: Refer to AMBA 3 AHB-Lite Protocol V1.0 for more information.

2.3.3. Interrupt interface

Table 2.7. Interrupt Port

Name	Type	Width	Description
IRQ	In	1~8	Peripheral interrupts
TIMER_IRQ_OUT	Out	1	Timer interrupt , exist only when “TIMER_ENABLE” checked
TIMER_IRQ_IN	In	1	Timer interrupt, exist only when “TIMER_ENABLE” unchecked

2.4. Attribute Summary

The configurable attributes of the RISC-V MC CPU IP are shown in Table 2.8 and are described in Table 2.9.
The attributes can be configured through the Propel Builder software.

Table 2.8. Configurable Attributes

Attribute	Selectable Values	Default	Dependency on Other Attributes
General			
SIMULATION	Checked, Unchecked	UnChecked	—
DEBUG_ENABLE	Checked, Unchecked	UnChecked	—
TIMER_ENABLE	Checked, Unchecked	UnChecked	—
PIC_ENABLE	Checked, Unchecked	UnChecked	—
PICTIMER_START_ADDR	0~0xFFFFFC00	0xFFFF0000	Enabled when PIC_ENABLE or TIMER_ENABLE
IRQ_NUM	2~8	8	Enabled when PIC_ENABLE
C_EXT	Checked, Unchecked	Unchecked	—
JTAG_CHANNEL	14~16	14	Enabled when DEBUG_ENABLE

Table 2.9. Attributes Description

Attribute	Description
SIMULATION	1: simulation mode for CPU 0: synthesis mode for CPU
DEBUG_ENABLE	1: debug function enable 0: debug function disable
TIMER_ENABLE	1: timer enable 0: timer disable
PIC_ENABLE	1: pic enable 0: pic disable
PICTIMER_START_ADDR	Start address of PIC and Timer
IRQ_NUM	Interrupt number for peripherals
C_EXT	1: support for compressed instruction 0: no support for compressed instruction
JTAG_CHANNEL	JTAG channel select

3. RISC-V MC CPU IP Generation

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

To generate the CPU IP Core module:

1. In Propel Builder, create a new design. Select the CPU package.
2. Enter the component name. Click **Next** (Figure 3.1).

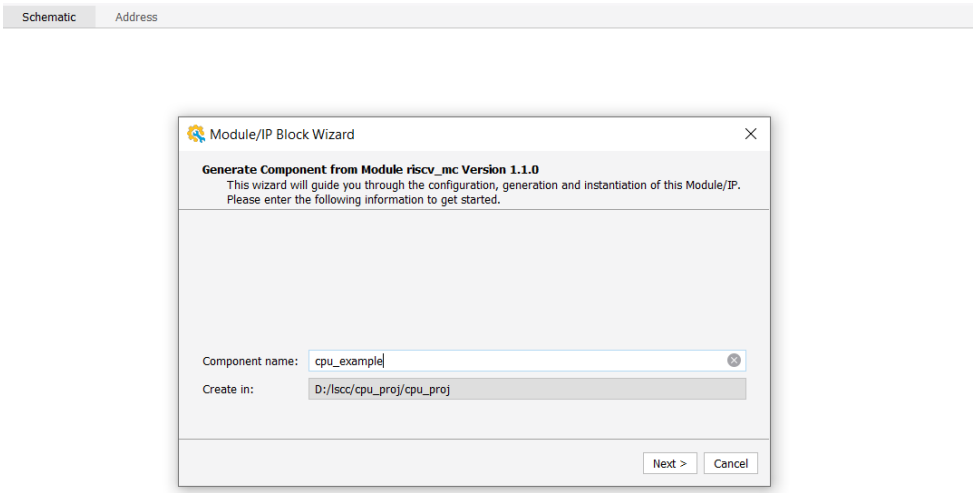

Figure 3.1. Entering Component Name

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

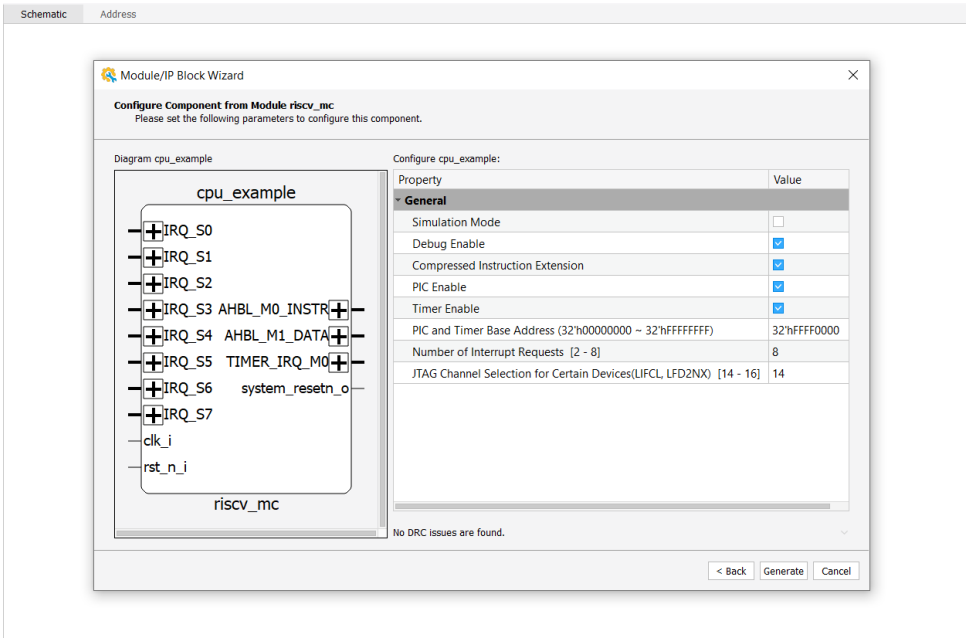

Figure 3.2. Configuring Parameters

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

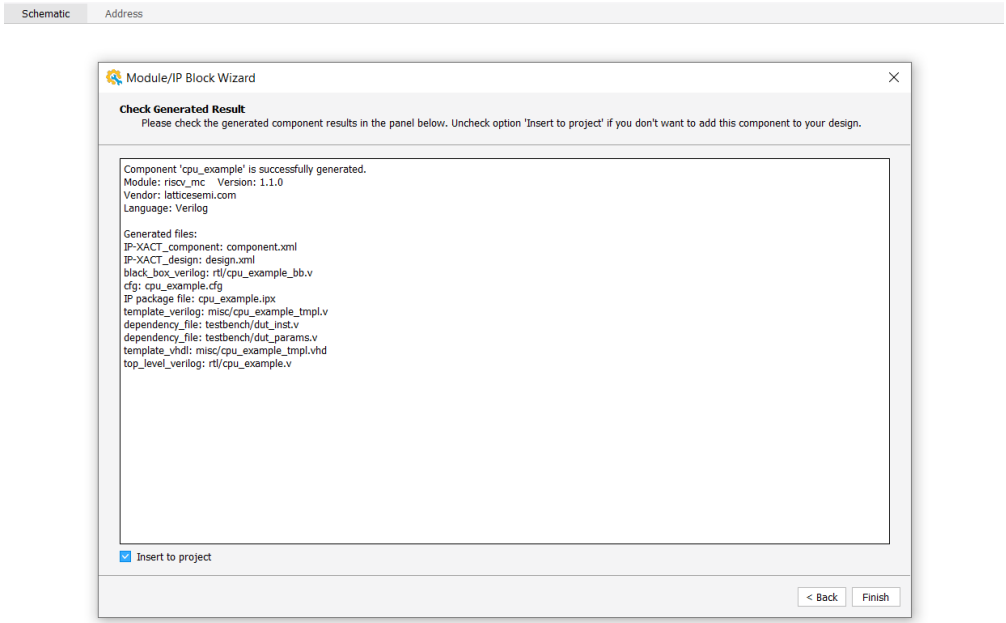

Figure 3.3. Verifying Results

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

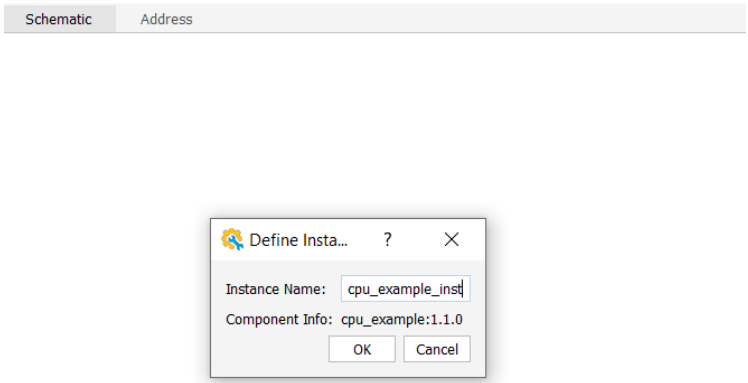

Figure 3.4. Specifying Instance name

The CPU IP instance is successfully generated (Figure 3.5).

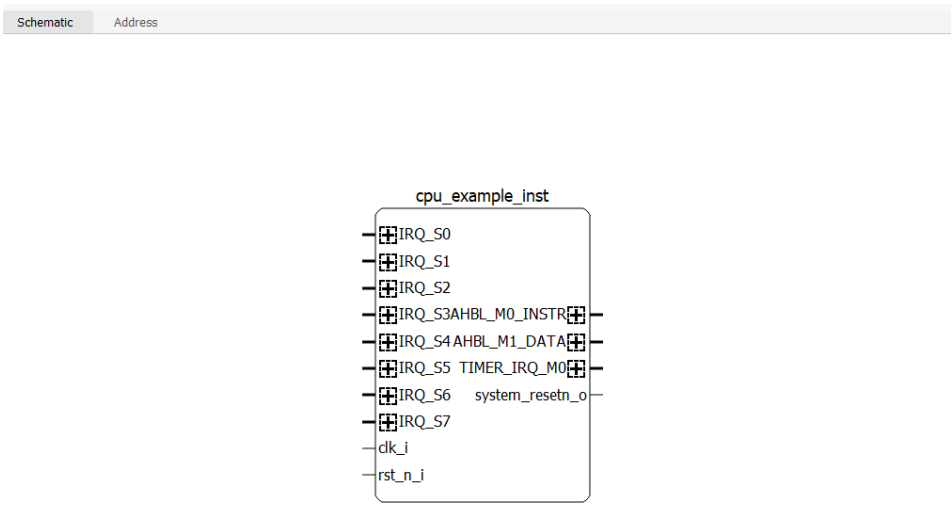

Figure 3.5. Generated Instance

Appendix A. Resource Utilization

Table A.3.1. Resource Utilization in MachXO3D Device

Configuration	LUTs	Registers	sysMEM EBRs
Processor core only	1507	809	4
Processor core + PIC	1640	851	4
Processor core + Timer	1857	955	4
Processor core + Debug	1716	1111	4
Processor core + C_EXT	1754	861	4
Processor core + PIC + Timer	1864	965	4
Processor core + PIC + Timer + Debug	2140	1268	4
Processor core + PIC + Timer + Debug + C_EXT	2410	1352	4

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

Table A.3.2. Resource Utilization in CrossLink-NX Device

Configuration	LUTs	Registers	sysMEM EBRs
Processor core only	1619	807	2
Processor core + PIC	1707	826	2
Processor core + Timer	2099	959	2
Processor core + Debug	1943	1182	2
Processor core + C_EXT	1893	852	2
Processor core + PIC + Timer	2110	963	2
Processor core + PIC + Timer + Debug	2489	1339	2
Processor core + PIC + Timer + Debug + C_EXT	2749	1352	2

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

References

- Lattice Semiconductor MachXO3D FPGA Web Page
- Lattice-Semiconductor CrossLink-NX FPGA Web Page
- Lattice Propel 1.0 User Guide (FPGA-UG-02110)
- Lattice Diamond Software 3.11 User Guide
- Lattice Radiant Software 2.1 User Guide
- AMBA 3 AHB-Lite Protocol V1.0
- RISC-V Privileged Specification (20190608)
- RISC-V Instruction Set Manual (20190608)
- Lattice Memory Mapped Interface (LMMI) and Lattice Interrupt Interface (LINTR) User Guide (FPGA-UG-02039)

Technical Support Assistance

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

Revision History

Revision 1.0, November 2020

Section	Change Summary
All	Initial release.

www.latticesemi.com