RISC-V SM CPU IP Core - Lattice Propel Builder 2.1

User Guide

FPGA-IPUG-02173-1.0

November 2021

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 associated risk the responsibility entirely of the 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
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

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 in Verilog HDL. It can be configured and generated using the Lattice Propel™ Builder software. It can be targeted to Certus™-NX/CertusPro™-NX/CrossLink™-NX/MachXO3D™/MachXO3™/MachXO2™ FPGA devices and implemented using the Lattice Diamond®/Radiant™ software Place and Route tool integrated with the Synplify Pro® synthesis tool.

1.1. Features

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

- RV32I 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 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 Propel)

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 SM CPU IP processes data and instructions while considering the external interrupts. As shown in Figure 2.1, the CPU IP has a 32-bit processor core and optional submodules. It uses one AHB-L interface (Read-Only) for instruction fetch and another AHB-L interface (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 multiplex run in system clock domain. The Core Debug runs in both system clock domain and JTAG clock domain.

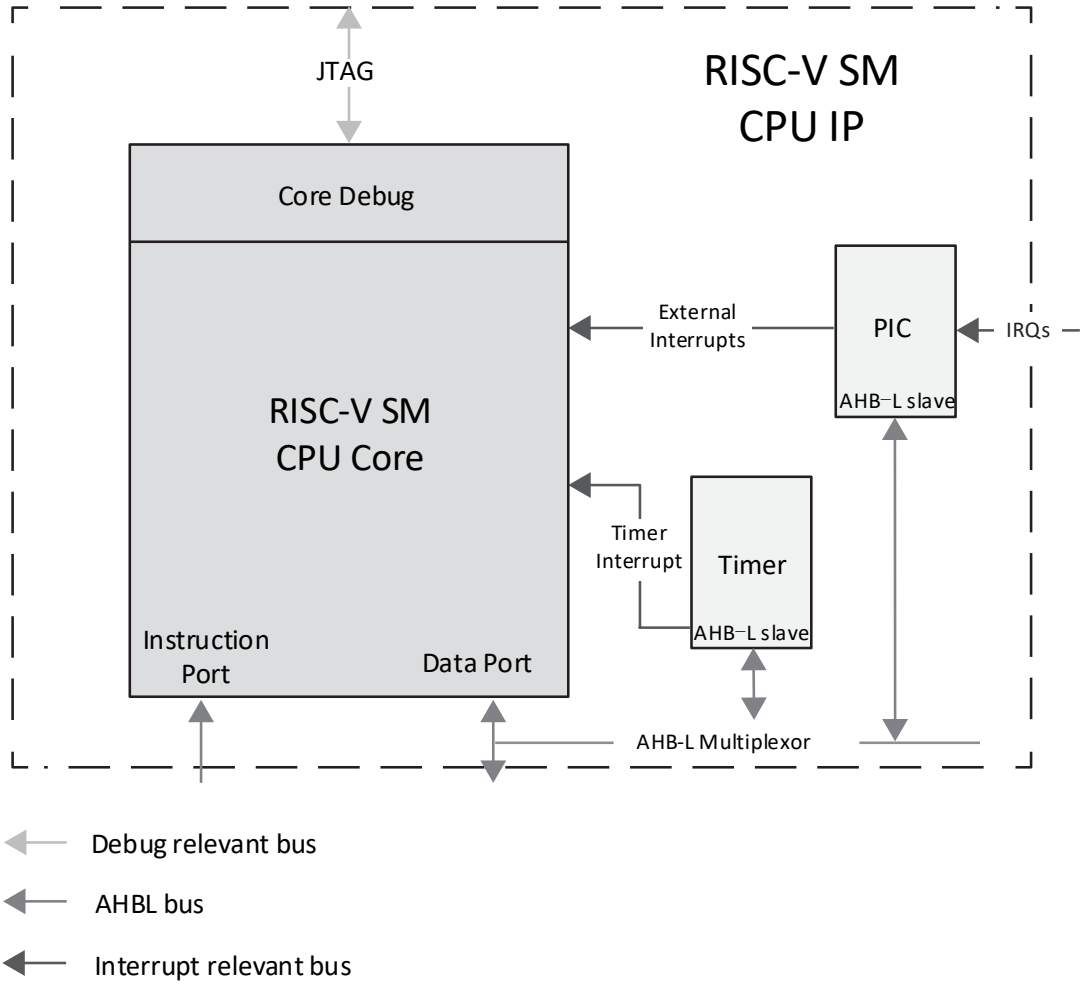

Figure 2.1. RISC-V SM CPU IP Diagram

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.

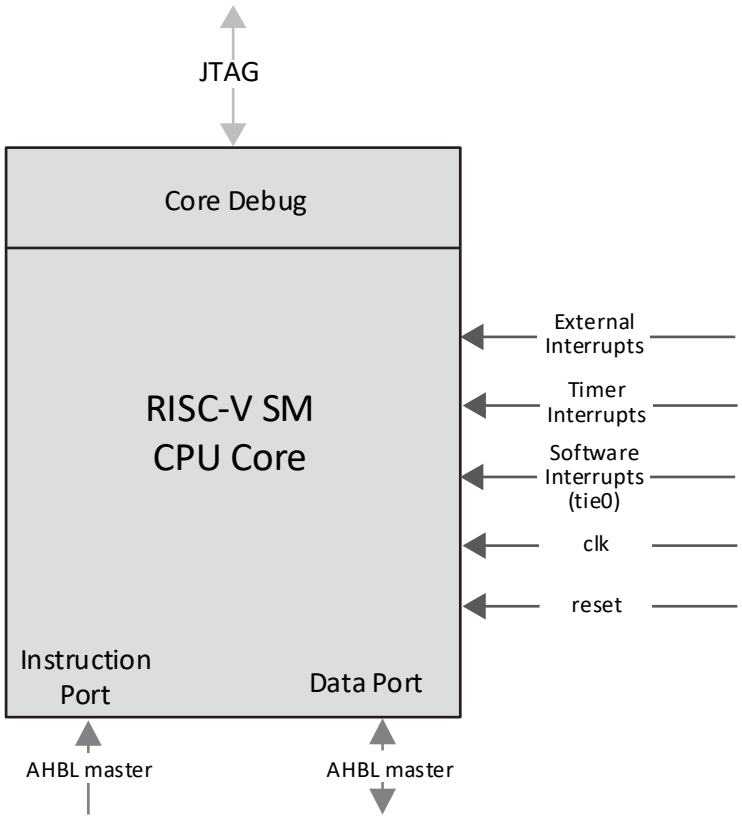

Figure 2.2. RISC-V SM CPU Core Block Diagram

2.2.1.1. Interrupt

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 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

Processor core enters into low power mode with *WFI* command, PC halts during low power mode, and the 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 SM CPU Core Control and Status Registers

CSR Number	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 (machine mode) in this CPU core. bit [7]: mpie, mie before entering trap, 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, machine mode external interrupt enable bit[7]: mtie, machine mode timer interrupt enable bit[3]: msie, machine mode software interrupt enable
0x305	mtvec (machine trap-vector base-address register)	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	mepc (machine exception program counter)	cannot be accessed	bits[31:0]: when trap is taken into machine 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)	cannot be accessed	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/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.

2.2.2. 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 2 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.

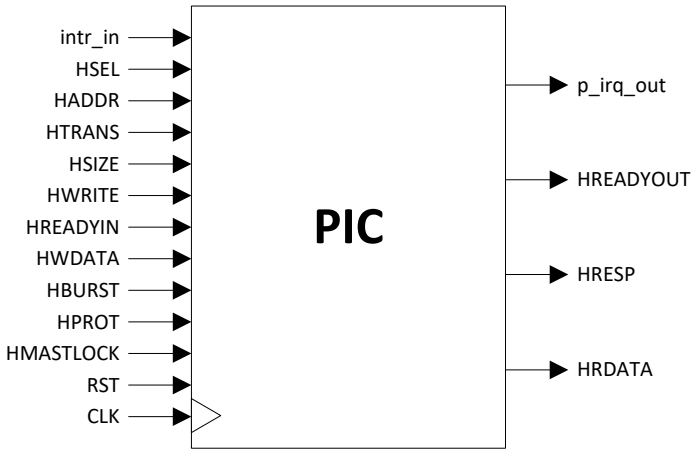

Figure 2.3. PIC Block Diagram

Table 2.2 provides the descriptions of PIC registers.

Table 2.2. PIC Registers

Offset	Name	Description
0x000	PIC_STATUS	Interrupt Status Register (read-write) (parameterizable width, min=2, max=8) Indicate the pending interrupt at corresponding interrupt request port(irq[i]). **Field / Name / Access / Width / Reset** [N-1] / PIC_STATUS [N-1] / RW / 1 / 0x0 … / … / … / … / … [1] / PIC_STATUS [1] / RW / 1 / 0x0 [0] / PIC_STATUS [0] / RW / 1 / 0x0 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]

<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, will get 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 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.

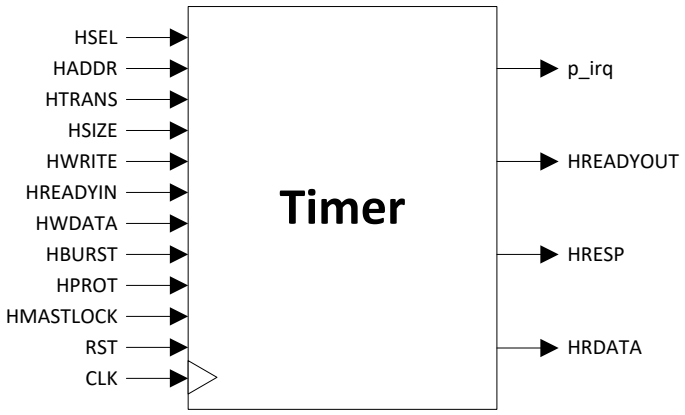

Figure 2.4. Timer Block Diagram

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><th>Field</th><th>Name</th><th>Access</th><th>Width</th><th>Reset</th></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><th>Field</th><th>Name</th><th>Access</th><th>Width</th><th>Reset</th></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 2.7 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

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	1	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	—
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	1	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	—

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_Sx	In	1~8	Peripheral interrupts
TIMER_IRQ_M0	Out	1	Timer interrupt output, exist only when "TIMER_ENABLE" checked
TIMER_IRQ_S0	In	1	Timer interrupt input, exist only when "TIMER_ENABLE" 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 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
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
JTAG_CHANNEL	JTAG channel select

3. RISC-V SM CPU IP Generation

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

To generate the CPU IP module:

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

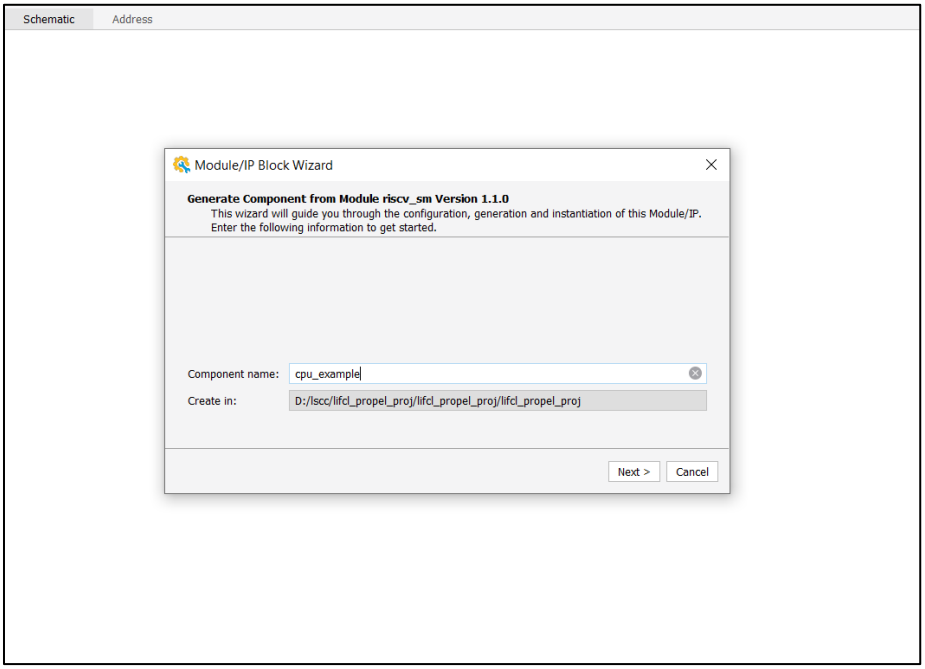

Figure 3.1. Entering Component Name

3. Configure the parameters as needed. Click **Generate**.

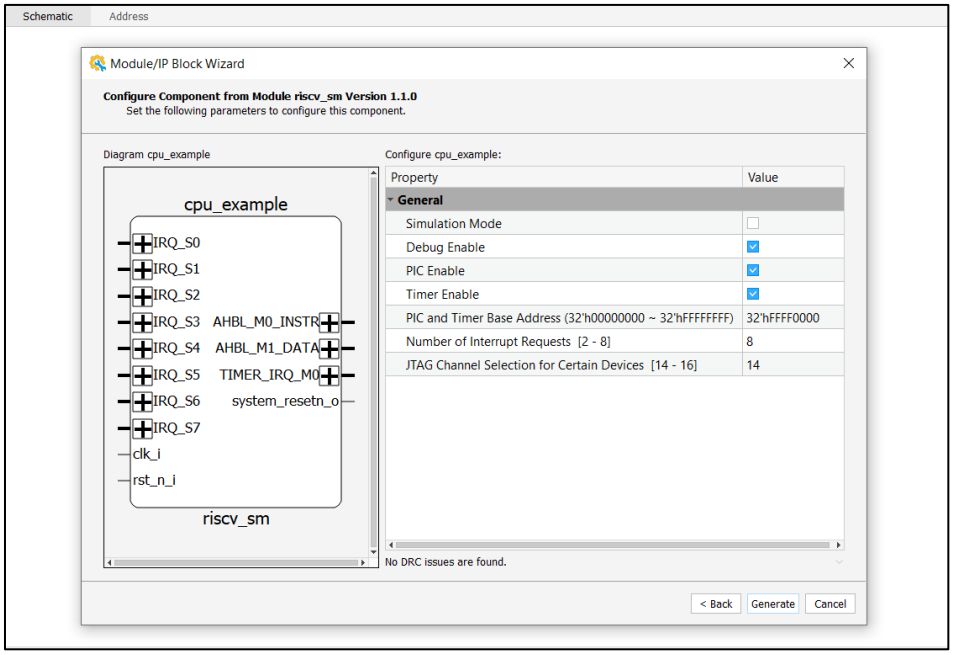

Figure 3.2. Configuring Parameters

4. Verify the information. Click **Finish**.

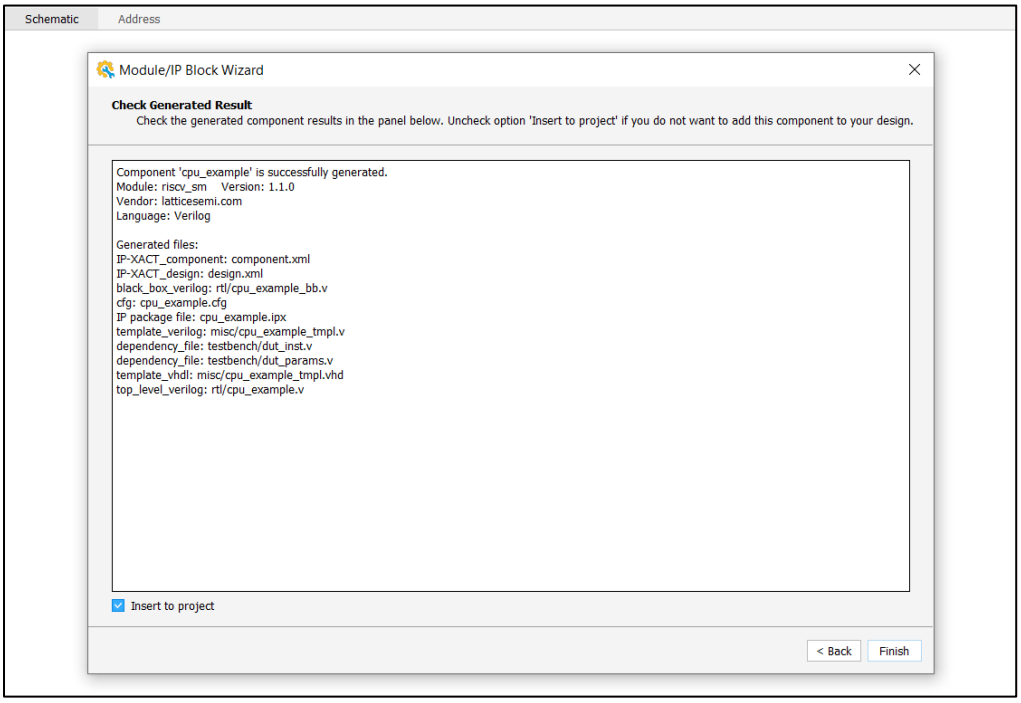

Figure 3.3. Verifying Results

5. Confirm or modify the module instance name. Click **OK**.

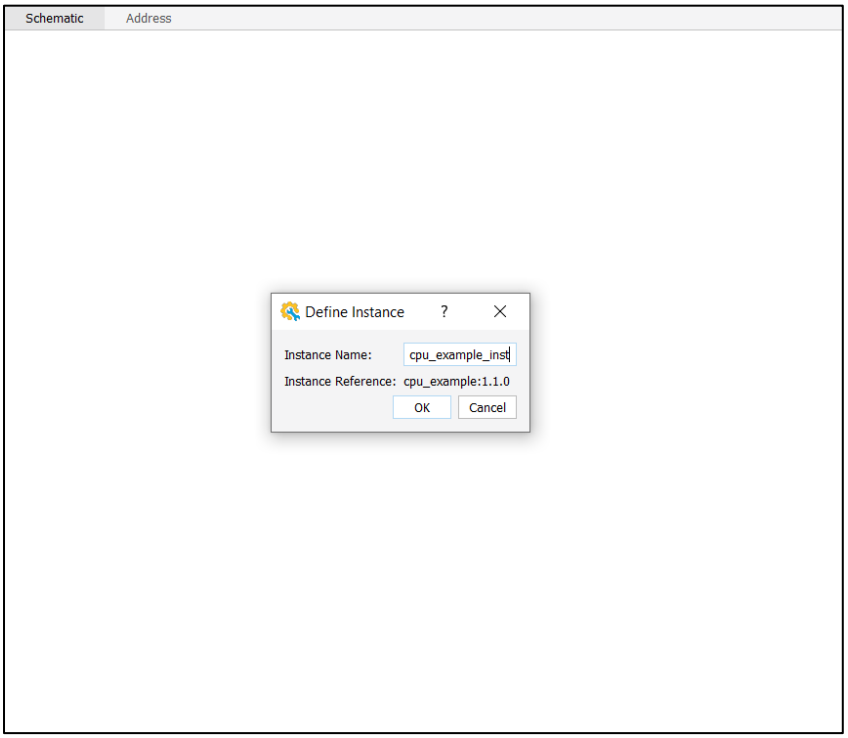

Figure 3.4. Specifying Instance name

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

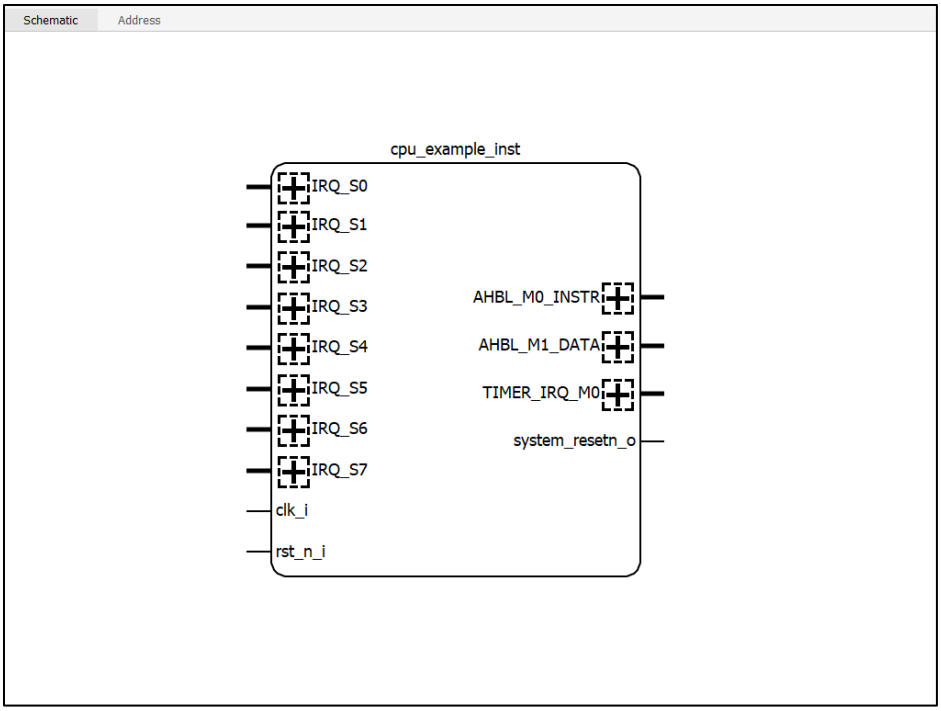

Figure 3.5. Generated Instance

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.

References

- Lattice Semiconductor CrossLink-NX FPGA Web Page
- Lattice Semiconductor MachXO3D FPGA Web Page
- Lattice Semiconductor MachXO3 FPGA Web Page
- Lattice Semiconductor MachXO2 FPGA Web Page
- Lattice Propel 2.0 User Guide
- Lattice Diamond Software 3.12 User Guide
- Lattice Radiant Software 3.0 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

Technical Support Assistance

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

Revision History

Document Revision 1.0, November 2021

Section	Change Summary
All	Initial release.

www.latticesemi.com