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

Acronyms in This Document ........................................................................................................... 5
1. Introduction ....................................................................................................................................... 6
  1.1. Quick Facts ................................................................................................................................. 6
  1.2. Features ...................................................................................................................................... 6
  1.3. Conventions ............................................................................................................................... 6
2. Functional Descriptions .................................................................................................................. 7
  2.1. Overview ..................................................................................................................................... 7
  2.2. Modules Description .................................................................................................................. 7
    2.2.1. RISC-V Processor Core ...................................................................................................... 7
    2.2.2. Submodule (PIC/Timer) ..................................................................................................... 10
  2.4. Signal Description ...................................................................................................................... 13
    2.4.1. Clock and Reset .................................................................................................................. 13
    2.4.2. Instruction and Data Interface ............................................................................................ 13
    2.4.3. Interrupt interface .............................................................................................................. 14
  2.5. Attribute Summary ..................................................................................................................... 14
3. RISC-V MC CPU IP Generation ...................................................................................................... 16
Appendix A. Resource Utilization ........................................................................................................ 19
References ........................................................................................................................................... 20
Technical Support Assistance ............................................................................................................. 21
Revision History ................................................................................................................................. 22
Figures

Figure 2.1. RISC-V MC Soft IP Diagram ................................................................. 7
Figure 2.2. RISC-V MC Processor Core Block Diagram .................................................. 8
Figure 2.3. PIC Block Diagram .................................................................................. 10
Figure 2.4. Timer Block Diagram .............................................................................. 12
Figure 3.1. Entering Component Name ....................................................................... 16
Figure 3.2. Configuring Parameters ............................................................................ 16
Figure 3.3. Verifying Results ...................................................................................... 17
Figure 3.4. Specifying Instance Name .......................................................................... 17
Figure 3.5. Generated Instance .................................................................................. 18

Tables

Table 1.1 RISC-V MC CPU IP Core Quick Facts .......................................................... 6
Table 2.1. RISC-V Processor Core Control and Status Registers ........................................ 9
Table 2.2. PIC Registers ............................................................................................ 10
Table 2.3. Timer Registers ......................................................................................... 12
Table 2.4. Clock and Reset Ports ............................................................................... 13
Table 2.5. Instruction Ports ....................................................................................... 13
Table 2.6. Data Ports ............................................................................................... 13
Table 2.7. Interrupt Ports ......................................................................................... 14
Table 2.8. Configurable Attributes ............................................................................ 14
Table 2.9. Attributes Description .............................................................................. 14
Table A.1. Resource Utilization in MachXO3D Device (with Cache Disabled) ............... 19
Table A.2. Resource Utilization in CrossLink-NX Device (with Cache Disabled) .......... 19
Table A.3. Resource Utilization in CrossLink-NX Device (with Cache Enabled) .......... 19
Acronyms in This Document
A list of acronyms used in this document.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHB-L</td>
<td>Advanced High-performance Bus – Lite</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CSR</td>
<td>Control and Status Register</td>
</tr>
<tr>
<td>DMIPS</td>
<td>Dhrystone MIPS (Million Instructions per Second)</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>GDB</td>
<td>Gnu Debugger</td>
</tr>
<tr>
<td>HDL</td>
<td>Hardware Description Language</td>
</tr>
<tr>
<td>IP</td>
<td>Intellectual Property</td>
</tr>
<tr>
<td>IRQ</td>
<td>Interrupt Request</td>
</tr>
<tr>
<td>ISA</td>
<td>Instruction Set Architecture</td>
</tr>
<tr>
<td>JTAG</td>
<td>Joint Test Action Group</td>
</tr>
<tr>
<td>LUT</td>
<td>Lookup-Table</td>
</tr>
<tr>
<td>MC</td>
<td>Micro-Controller (RISC-V for Micro-Controller applications)</td>
</tr>
<tr>
<td>OpenOCD</td>
<td>Open On-Chip Debugger</td>
</tr>
<tr>
<td>PIC</td>
<td>Programmable Interrupt Controller</td>
</tr>
<tr>
<td>RISC-V</td>
<td>Reduced instruction set computer-V (five)</td>
</tr>
<tr>
<td>RV32IMC</td>
<td>RISC-V Integer, M and Compressed Instruction Sets</td>
</tr>
<tr>
<td>WFI</td>
<td>Wait For Interrupt</td>
</tr>
</tbody>
</table>
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 is with instruction and data caches. The CPU core supports RV32IMC instruction set, external interrupts, and debug feature that 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 Lattice Avant™, MachXO5™-NX, Certus™-NX, CertusPro™-NX, CrossLink™-NX, MachXO3™, MachXO3L™, MachXO3LF™, MachXO2™ FPGA devices.

1.1. Quick Facts

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

<table>
<thead>
<tr>
<th>IP Requirements</th>
<th>Supported FPGA Family</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Utilization</td>
<td>Avant, MachXO5-NX, Certus-NX, CertusPro-NX, CrossLink-NX, MachXO3D, MachXO3L™, MachXO3LF™, MachXO2</td>
</tr>
<tr>
<td>Targeted Devices</td>
<td>LAV-AT, LFMXOS, LFD2NX, LFCPNX, LIFCL, LAMXO3D, LCMXO3L, LCMXO3LF, LCMXO2</td>
</tr>
<tr>
<td>Supported User Interfaces</td>
<td>AHB – Lite Interface</td>
</tr>
<tr>
<td>Resources</td>
<td>See Table A.1, Table A.2, Table A.3.</td>
</tr>
</tbody>
</table>

1.2. Features

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

- RV32IMC instruction set
- Five stages of pipelines
- Support the AHB-L bus standard for instruction/data port
- Optional caches, including a 4 KB two-way instruction cache and a 4 KB two-way data cache (for Avant, MachXO5-NX, Certus-NX, CertusPro-NX, and CrossLink-NX only)
- Optional debug using Gnu Debugger (GDB) and Open On-Chip Debugger (OpenOCD)
- Optional PIC module
- Optional Timer module
- Interrupt and exception handling under Machine Mode
- > 0.7 DMIPS/MHz performance
- > 100 MHz on CrossLink-NX device (tested on the Hello World template provided by Propel)

1.3. Conventions

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

2.1. Overview
The RISC-V MC CPU IP processes data and instructions while monitoring 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 MC Soft IP Diagram](image)

2.2. Modules Description

2.2.1. RISC-V Processor Core
The processor core follows the RV32IMC instruction set. Figure 2.2 shows the 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 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 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.

Note: This function is not supported for Avant devices at this release.

2.2.1.5. Instruction and Data Caches
The processor core supports instruction and data caches.

The instruction and data caches are both 4 KB two-way set associative, each cache line contains 32 bytes. The cache strategy for data cache is write through, and the cache eviction policy of both caches is round robin.

The Instruction and data caches can be enabled/disabled together by checking/unchecked the “CACHE_ENABLE” option when instantiating CPU in Propel, and the cacheable address range can be configured by setting the “CACHEABLE_ADDR_LOW” and “CACHEABLE_ADDR_HIGH” parameters.
It should be noted that the instruction and data caches should only be enabled for Certus-NX, CertusPro-NX, CrossLink-NX and MachXO5-NX devices, as they have enough resources to support the caches.

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 Processor Core Control and Status Registers

<table>
<thead>
<tr>
<th>CSR No.</th>
<th>CSR Name</th>
<th>Access</th>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x300</td>
<td>mstatus (machine status register)</td>
<td>read/write</td>
<td>bit[12:11]: mpp, privilege mode before entering trap, should always be 2'b11 (machine mode) in this CPU core.  &lt;br&gt;bit[7]: mpie, mie before entering trap, updated to mie value when entering to trap.  &lt;br&gt;bit[3]: mie, global interrupt enable.</td>
</tr>
<tr>
<td>0x304</td>
<td>mie (machine interrupt enable register)</td>
<td>read/write</td>
<td>bit[11]: meie, machine mode external interrupt enable.  &lt;br&gt;bit[7]: mtie, machine mode timer interrupt enable.  &lt;br&gt;bit[3]: msie, machine mode software interrupt enable.</td>
</tr>
<tr>
<td>0x305</td>
<td>mtvec (initialized to 0x20)</td>
<td>read/write</td>
<td>bit[31:2]: trap vector base address, 4-byte aligned.  &lt;br&gt;bit[0]: trap vector mode, all traps set the program counter to the base address in RISC-V MC CPU core. Bit[1] is not supported. Only 1'b0 - direct mode and 1'b1 - vectored mode are available.</td>
</tr>
<tr>
<td>0x340</td>
<td>mscratch</td>
<td>read/write</td>
<td>bit[31:0]: for use by machine mode, it is used to hold a pointer to a machine-mode hart-local context space and swapped with a user register upon entry to a machine mode trap handler.</td>
</tr>
<tr>
<td>0x341</td>
<td>mepc (machine exception program counter)</td>
<td>read/write</td>
<td>bit[31:0]: when trap is taken into machine mode, mepc is used to store the address of the instruction that encountered exception.</td>
</tr>
<tr>
<td>0x342</td>
<td>mcause (machine cause register)</td>
<td>read only</td>
<td>bit[31]: 1'b1 - interrupt, 1'b0 - exception  &lt;br&gt;bit[3:0]: exception code for interrupt:  &lt;br&gt;3 - machine software interrupt  &lt;br&gt;7 - machine timer interrupt  &lt;br&gt;11 - machine external interrupt  &lt;br&gt;for exception:  &lt;br&gt;0 - instruction address misaligned  &lt;br&gt;1 - instruction access fault  &lt;br&gt;2 - illegal instruction  &lt;br&gt;4 - load address misaligned  &lt;br&gt;5 - load access fault</td>
</tr>
<tr>
<td>0x343</td>
<td>mtval (machine trap value register)</td>
<td>read only</td>
<td>bit[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.</td>
</tr>
<tr>
<td>0x344</td>
<td>mip (machine interrupt pending register)</td>
<td>read/write</td>
<td>bit[11]: meip, machine mode external interrupt pending, read only.  &lt;br&gt;bit[7]: mtip, machine mode timer interrupt pending, read only.  &lt;br&gt;bit[3]: msip, machine mode software interrupt pending, readable and writable.</td>
</tr>
</tbody>
</table>
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 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](image)

Table 2.2 provides the descriptions of PIC registers.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>PIC_STATUS</td>
<td>Interrupt Status Register (read-write) (parameterizable width, min=2, max=8)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Indicate the pending interrupt at corresponding interrupt request port(irq[i] at top level).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Name</th>
<th>Access</th>
<th>Width</th>
<th>Reset</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</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]
<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Description</th>
<th>Field</th>
<th>Name</th>
<th>Access</th>
<th>Width</th>
<th>Reset</th>
</tr>
</thead>
<tbody>
<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]).</td>
<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>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[1]</td>
<td>PIC_ENABLE[1]</td>
<td>RW</td>
<td>1</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[0]</td>
<td>PIC_ENABLE[0]</td>
<td>RW</td>
<td>1</td>
<td>0x0</td>
</tr>
</tbody>
</table>

PIC_ENABLE[i]:
Read
- 0 – irq[i] disabled
- 1 – irq[i] enabled
Write
- 0 – disable irq[i]
- 1 – enable irq[i]

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Description</th>
<th>Field</th>
<th>Name</th>
<th>Access</th>
<th>Width</th>
<th>Reset</th>
</tr>
</thead>
<tbody>
<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]).</td>
<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>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[1]</td>
<td>PIC_SET[1]</td>
<td>W</td>
<td>1</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[0]</td>
<td>PIC_SET[0]</td>
<td>W</td>
<td>1</td>
<td>0x0</td>
</tr>
</tbody>
</table>

PIC_SET[i]:
Read
- Invalid operation gets 0.
Write
- 0 – no effect
- 1 – set interrupt status for irq[i] (set PIC_STATUS[i])

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Description</th>
<th>Field</th>
<th>Name</th>
<th>Access</th>
<th>Width</th>
<th>Reset</th>
</tr>
</thead>
<tbody>
<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.</td>
<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>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[1]</td>
<td>PIC_POL[1]</td>
<td>RW</td>
<td>1</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[0]</td>
<td>PIC_POL[0]</td>
<td>RW</td>
<td>1</td>
<td>0x0</td>
</tr>
</tbody>
</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

**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 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 an AHB-L interface, as shown in Figure 2.4.

Table 2.3 provides the descriptions of Timer registers.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x400</td>
<td>TIMER_CNT_L</td>
<td>Lower 32 bits of Timer counter register.</td>
</tr>
<tr>
<td></td>
<td>Field Name</td>
<td>Access</td>
</tr>
<tr>
<td>[63:0]</td>
<td>mtime</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td>mttime</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Field Name</td>
<td>Access</td>
</tr>
<tr>
<td>[63:0]</td>
<td>mtimecmp</td>
<td>RW</td>
</tr>
<tr>
<td></td>
<td>mtimecmp</td>
<td>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. The interrupt remains posted until mtimecmp becomes greater than mtime (typically as a result of writing mtimecmp).</td>
</tr>
</tbody>
</table>
2.4. Signal Description

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

2.4.1. Clock and Reset

Table 2.4. Clock and Reset Ports

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk_i</td>
<td>In</td>
<td>1</td>
<td>RISC-V soft IP clock</td>
</tr>
<tr>
<td>rst_n_i</td>
<td>In</td>
<td>1</td>
<td>Global reset (active low)</td>
</tr>
<tr>
<td>system_resetn_o</td>
<td>Out</td>
<td>1</td>
<td>Combined Global reset and Debug Reset from JTAG</td>
</tr>
</tbody>
</table>

2.4.2. Instruction and Data Interface

Table 2.5. Instruction Ports

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHBL_M0_INSTR - HADDR</td>
<td>Out</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>AHBL_M0_INSTR - HWRITE</td>
<td>Out</td>
<td>1</td>
<td>Fixed to 1’b0</td>
</tr>
<tr>
<td>AHBL_M0_INSTR - HSIZE</td>
<td>Out</td>
<td>3</td>
<td>Fixed to 3’b010</td>
</tr>
<tr>
<td>AHBL_M0_INSTR - HPROT</td>
<td>Out</td>
<td>4</td>
<td>Fixed to 4’b1110 when cache is not enabled</td>
</tr>
<tr>
<td>AHBL_M0_INSTR - HTRANS</td>
<td>Out</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>AHBL_M0_INSTR - HBURST</td>
<td>Out</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AHBL_M0_INSTR - HMASTLOCK</td>
<td>Out</td>
<td>1</td>
<td>Fixed to 1’b0</td>
</tr>
<tr>
<td>AHBL_M0_INSTR - HWDATA</td>
<td>Out</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>AHBL_M0_INSTR - HRDATA</td>
<td>In</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>AHBL_M0_INSTR - HREADY</td>
<td>In</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>AHBL_M0_INSTR - HRESP</td>
<td>In</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.6. Data Ports

<table>
<thead>
<tr>
<th>Name</th>
<th>Direction</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHBL_M1_DATA - HADDR</td>
<td>Out</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>AHBL_M1_DATA - HWRITE</td>
<td>Out</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>AHBL_M1_DATA - HSIZE</td>
<td>Out</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AHBL_M1_DATA - HPROT</td>
<td>Out</td>
<td>4</td>
<td>Fixed to 4’b1111 when cache is not enabled</td>
</tr>
<tr>
<td>AHBL_M1_DATA - HTRANS</td>
<td>Out</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>AHBL_M1_DATA - HBURST</td>
<td>Out</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>AHBL_M1_DATA - HMASTLOCK</td>
<td>Out</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>AHBL_M1_DATA - HSEL</td>
<td>Out</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>AHBL_M1_DATA - HWDATA</td>
<td>Out</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>AHBL_M1_DATA - HRDATA</td>
<td>In</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>AHBL_M1_DATA - HREADY</td>
<td>In</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

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

Table 2.7. Interrupt Ports

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Width</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRQ_Sx</td>
<td>In</td>
<td>1~8</td>
<td>Peripheral interrupts.</td>
</tr>
<tr>
<td>TIMER_IRQ_M0</td>
<td>Out</td>
<td>1</td>
<td>Timer interrupt output, exists only when &quot;TIMER_ENABLE&quot; is checked.</td>
</tr>
<tr>
<td>TIMER_IRQ_S0</td>
<td>In</td>
<td>1</td>
<td>Timer interrupt input, exists only when &quot;TIMER_ENABLE&quot; is unchecked.</td>
</tr>
</tbody>
</table>

2.5. 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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Selectable Values</th>
<th>Default</th>
<th>Dependency on Other Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIMULATION</td>
<td>Checked, Unchecked</td>
<td>UnChecked</td>
<td>—</td>
</tr>
<tr>
<td>CACHE_ENABLE</td>
<td>Checked, Unchecked</td>
<td>Checked</td>
<td>Disabled when SIMULATION</td>
</tr>
<tr>
<td>ICACHE_RANGE_LOW</td>
<td>0 ~ 0xFFFFFFFF</td>
<td>0</td>
<td>Enabled when CACHE_ENABLE</td>
</tr>
<tr>
<td>ICACHE_RANGE_HIGH</td>
<td>0 ~ 0xFFFFFFFF</td>
<td>0x00000000</td>
<td>Enabled when CACHE_ENABLE</td>
</tr>
<tr>
<td>DCACHE_RANGE_LOW</td>
<td>0 ~ 0xFFFFFFFF</td>
<td>0</td>
<td>Enabled when CACHE_ENABLE</td>
</tr>
<tr>
<td>DCACHE_RANGE_HIGH</td>
<td>0 ~ 0xFFFFFFFF</td>
<td>0x00000000</td>
<td>Enabled when CACHE_ENABLE</td>
</tr>
<tr>
<td>DEBUG_ENABLE</td>
<td>Checked, Unchecked</td>
<td>Checked</td>
<td>Disabled for Avant devices</td>
</tr>
<tr>
<td>TIMER_ENABLE</td>
<td>Checked, Unchecked</td>
<td>Checked</td>
<td>—</td>
</tr>
<tr>
<td>PIC_ENABLE</td>
<td>Checked, Unchecked</td>
<td>Checked</td>
<td></td>
</tr>
<tr>
<td>PICTIMER_START_ADDR</td>
<td>0 ~ 0xFFFFFB00</td>
<td>0xFFFF0000</td>
<td>Enabled when PIC_ENABLE or TIMER_ENABLE</td>
</tr>
<tr>
<td>IRQ_NUM</td>
<td>2 ~ 8</td>
<td>8</td>
<td>Enabled when PIC_ENABLE</td>
</tr>
<tr>
<td>C_EXT</td>
<td>Checked, Unchecked</td>
<td>Checked</td>
<td>—</td>
</tr>
<tr>
<td>M_EXT</td>
<td>Checked, Unchecked</td>
<td>Unchecked</td>
<td>Enabled when C_EXT (For Certus-NX, CertusPro-NX, CrossLink-NX, MachXO5-NX devices only.)</td>
</tr>
<tr>
<td>JTAG_CHANNEL</td>
<td>14~16</td>
<td>14</td>
<td>Enabled when DEBUG_ENABLE</td>
</tr>
</tbody>
</table>

Table 2.9. Attributes Description

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIMULATION</td>
<td>1: simulation mode 0: synthesis mode</td>
</tr>
<tr>
<td>ICACHE_ENABLE</td>
<td>1: enable instruction cache 0: disable instruction cache</td>
</tr>
<tr>
<td>DCACHE_ENABLE</td>
<td>1: enable data cache 0: disable data cache</td>
</tr>
<tr>
<td>ICACHE_RANGE_LOW</td>
<td>Lower limit of cacheable address range for instruction cache, this address itself is included.</td>
</tr>
<tr>
<td>ICACHE_RANGE_HIGH</td>
<td>Higher limit of cacheable address range for instruction cache, this address itself is included.</td>
</tr>
<tr>
<td>DCACHE_RANGE_LOW</td>
<td>Lower limit of cacheable address range for data cache, this address itself is included.</td>
</tr>
<tr>
<td>DCACHE_RANGE_HIGH</td>
<td>Higher limit of cacheable address range for data cache, this address itself is included.</td>
</tr>
<tr>
<td>DEBUG_ENABLE</td>
<td>1: debug function enable 0: debug function disable</td>
</tr>
<tr>
<td>TIMER_ENABLE</td>
<td>1: timer enable</td>
</tr>
<tr>
<td>Attribute</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>PIC_ENABLE</td>
<td>0: timer disable</td>
</tr>
<tr>
<td></td>
<td>1: pic enable</td>
</tr>
<tr>
<td></td>
<td>0: pic disable</td>
</tr>
<tr>
<td>PICTIMER_START_ADDR</td>
<td>Start address of PIC and Timer</td>
</tr>
<tr>
<td>IRQ_NUM</td>
<td>Number of Peripheral Interrupts</td>
</tr>
<tr>
<td>C_EXT</td>
<td>1: support for compressed instruction</td>
</tr>
<tr>
<td></td>
<td>0: no support for compressed instruction</td>
</tr>
<tr>
<td>M_EXT</td>
<td>1: support for M standard extension</td>
</tr>
<tr>
<td></td>
<td>0: no support for M standard extension</td>
</tr>
<tr>
<td>JTAG_CHANNEL</td>
<td>JTAG channel select</td>
</tr>
</tbody>
</table>

Notes:
- “ICACHE_ENABLE” and “DCACHE_ENABLE” should only be enabled when using CrossLink-NX Devices with sufficient resources. For MachXO2, MachXO3L, MachXO3LF and MachXO3D devices, cache features are not supported as their resources are limited.
- “ICACHE_RANGE_LOW” should not be larger than “ICACHE_RANGE_HIGH”, and address range between “ICACHE_RANGE_LOW” and “ICACHE_RANGE_HIGH” should not be overlapped with peripheral devices’ address ranges.
- Similarly, “DCACHE_RANGE_LOW” should not be larger than “DCACHE_RANGE_HIGH”, and address range between “DCACHE_RANGE_LOW” and “DCACHE_RANGE_HIGH” should not be overlapped with peripheral devices address ranges.
- “DEBUG_ENABLE” is not editable for Avant devices.
- “SIMULATION” and “DEBUG_ENABLE” cannot be enabled at the same time.
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 as shown in Figure 3.1. Click **Next**.

![Figure 3.1. Entering Component Name](image)

3. Configure the parameters as shown in Figure 3.2. Click **Generate**.

![Figure 3.2. Configuring Parameters](image)
4. Verify the information. Click **Finish**.

![Figure 3.3. Verifying Results](image_url)

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

![Figure 3.4. Specifying Instance Name](image_url)

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 MachXO3D Device (with Cache Disabled)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>LUTs</th>
<th>Registers</th>
<th>EBRs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor core only</td>
<td>1450</td>
<td>806</td>
<td>4</td>
</tr>
<tr>
<td>Processor core + PIC</td>
<td>1559</td>
<td>845</td>
<td>4</td>
</tr>
<tr>
<td>Processor core + Timer</td>
<td>1844</td>
<td>955</td>
<td>4</td>
</tr>
<tr>
<td>Processor core + Debug</td>
<td>1726</td>
<td>1108</td>
<td>4</td>
</tr>
<tr>
<td>Processor core + C_EXT</td>
<td>1738</td>
<td>860</td>
<td>4</td>
</tr>
<tr>
<td>Processor core + PIC + Timer</td>
<td>1927</td>
<td>989</td>
<td>4</td>
</tr>
<tr>
<td>Processor core + PIC + Timer + Debug</td>
<td>2162</td>
<td>1287</td>
<td>4</td>
</tr>
<tr>
<td>Processor core + PIC + Timer + Debug + C_EXT</td>
<td>2385</td>
<td>1345</td>
<td>4</td>
</tr>
</tbody>
</table>

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

### Table A.2. Resource Utilization in CrossLink-NX Device (with Cache Disabled)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>LUTs</th>
<th>Registers</th>
<th>EBRs</th>
<th>DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor core only</td>
<td>1620</td>
<td>821</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Processor core + PIC</td>
<td>1789</td>
<td>859</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Processor core + Timer</td>
<td>2103</td>
<td>959</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Processor core + Debug</td>
<td>1979</td>
<td>1194</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Processor core + C_EXT</td>
<td>1911</td>
<td>830</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Processor core + C_EXT + M_EXT</td>
<td>2417</td>
<td>1172</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Processor core + PIC + Timer</td>
<td>2226</td>
<td>994</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Processor core + PIC + Timer + Debug</td>
<td>2494</td>
<td>1375</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Processor core + PIC + Timer + Debug + C_EXT</td>
<td>2790</td>
<td>1450</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

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

### Table A.3. Resource Utilization in CrossLink-NX Device (with Cache Enabled)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>LUTs</th>
<th>Registers</th>
<th>EBRs</th>
<th>DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor core + only</td>
<td>3372</td>
<td>1423</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>Processor core + PIC</td>
<td>3568</td>
<td>1479</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>Processor core + Timer</td>
<td>3753</td>
<td>1574</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>Processor core + Debug</td>
<td>3687</td>
<td>1810</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>Processor core + C_EXT</td>
<td>3711</td>
<td>1551</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>Processor core + C_EXT + M_EXT</td>
<td>4133</td>
<td>1879</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>Processor core + PIC + Timer</td>
<td>3894</td>
<td>1606</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>Processor core + PIC + Timer + Debug</td>
<td>4309</td>
<td>1980</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>Processor core + PIC + Timer + Debug + C_EXT</td>
<td>4568</td>
<td>2032</td>
<td>16</td>
<td>0</td>
</tr>
</tbody>
</table>

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

- Lattice Propel Builder 2022.1 User Guide (FPGA-UG-02177)
- 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

### Revision 1.0, October 2022

<table>
<thead>
<tr>
<th>Section</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Production release.</td>
</tr>
</tbody>
</table>