Mailbox IP – Lattice Propel Builder 2025.2

IP Version: v1.0.0

User Guide

FPGA-IPUG-02306-1.0

December 2025

Disclaimers

Inclusive Language

This document was created consistent with Lattice Semiconductor's inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice's inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
AXI-Lite	Advanced eXtensible Interface – Lite
AHB-Lite	Advanced High-Performance Bus – Lite
APB	Advanced Peripheral Bus
CPU	Central Processing Unit
EBR	Embedded Block RAM
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GPIO	General Purpose Input/Output
GUI	Graphic User Interface
HDL	Hardware Description Language
IOPMP	I/O Physical Memory Protection
IP	Intellectual Property
IRQ	Interrupt Request
LUT	Look Up Table
MC	Micro-Controller (RISC-V for Micro-Controller applications)
PIC	Programmable Interrupt Controller
RAM	Random Access Memory
RISC-V	Reduced Instruction Set Computer-V (Five)
SoC	System-on-Chip
UART	Universal Asynchronous Receiver/Transmitter
W1C	Write One Clear

1. Introduction

The Lattice Semiconductor Mailbox IP is designed for a multi-processor environment that requires sharing data between each processor. The mailbox IP provides a bi-directional communication interface between two processors. The IP can be connected to the processor through AHB-Lite or AXI-Lite interfaces.

The Mailbox IP is implemented using Verilog HDL and it can be configured and generated using the Lattice Propel™ Builder software. The IP supports Certus™-N2, Lattice Avant™, MachXO5™-NX, CrossLinkU™-NX, CrossLink™-NX, CertusPro™-NX, and Certus-NX FPGA devices.

This document includes the description of a new example design that incorporates the Mailbox IP and can be generated using the RISC-V MC Multi-Processor Project template available in Lattice Propel 2025.2. Refer to the Example Design section for details.

1.1. Quick Facts

Table 1.1 presents a summary of the Mailbox IP.

Table 1.1. Mailbox IP Quick Facts

IP Requirements	Supported Devices	Certus-N2, Lattice Avant, MachXO5-NX, CrossLinkU-NX, CrossLink-NX, CertusPro-NX, and Certus-NX
Resource Utilization	Supported User Interfaces	AXI-Lite Interface and AHB-Lite Interface
	Resources	See Table A.1 and Table A.2.
Design Tool Support	Lattice Implementation	IP v1.0.0 – Lattice Propel Builder 2025.2, Lattice Radiant™ Software 2025.2
	Simulation	For a list of supported simulators, see the Lattice Radiant software user guide.

1.2. Features

The Mailbox IP has the following features:

- AXI-Lite interface and AHB-Lite interface
- Configurable depth of mailbox
- Configurable interrupt thresholds and maskable interrupts

1.3. Conventions

1.3.1. Nomenclature

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

1.3.2. Signal Names

Signal Names that end with:

- _n are active low.
- _i are input signals.
- _o are output signals.
- _io are bi-directional input/output signals.

1.4. Licensing and Ordering Information

The Mailbox 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 Mailbox IP is used for multi-processor system-on-chips (SoCs) for bi-directional communication between two processors or two subsystems. Figure 2.1. Mailbox IP Diagram

shows a typical connection for the AHB-Lite system to share data between CPU1 and CPU2. You can configure the interrupt threshold and the Mailbox IP can notify the processors to start to receive data or stop sending data. The data flow is FIFO-based. You may select the FIFO implementation and FIFO depth through configurable attributes.

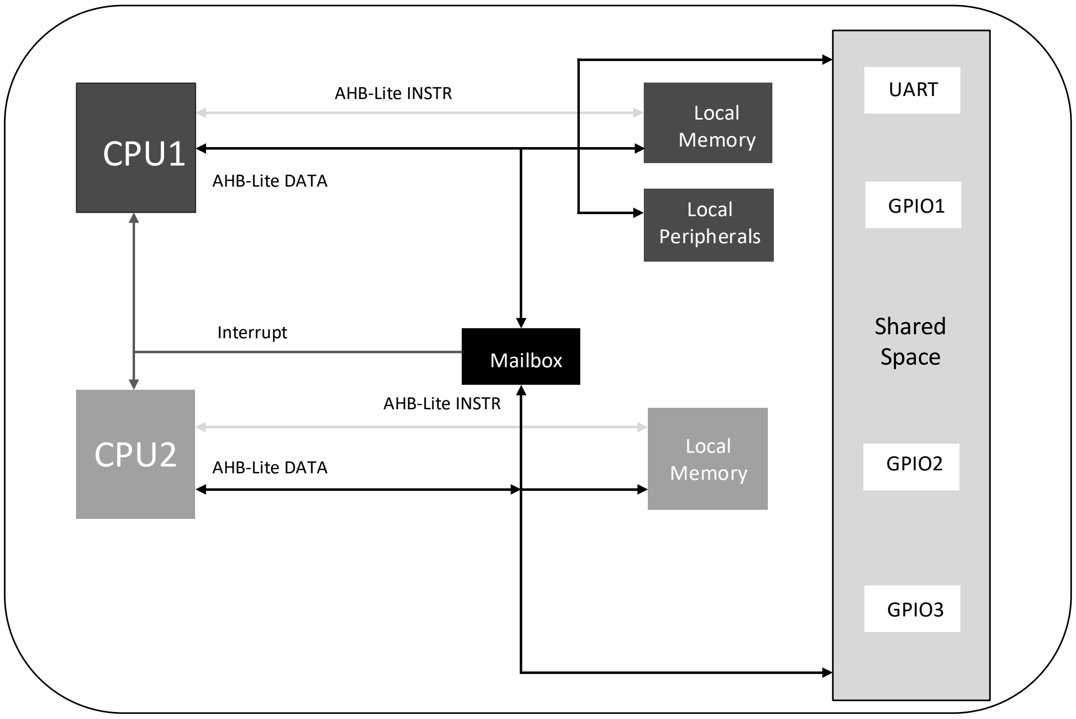


Figure 2.1. Mailbox IP Diagram

2.2. Modules Description

2.2.1. Bus Interface

The Mailbox IP has two data bus interfaces to access internal memory-mapped registers. These interfaces need to be connected to different processors in a multi-processor system. Each interface can be configured as AXI-Lite or AHB-Lite.

2.2.2. Registers

- Read and Write Data Registers: Processors can send data to or receive data from these registers. The registers are FIFO-based, and the depth can be configured through attributes.
- Status Registers: Processors can access these registers to get the status of the IP.
- Interrupt Registers: Processors can handle the interrupt behaviors, such as interrupt thresholds, mask, and interrupts status through these registers.

Table 2.1. Register Memory Map

Address	Register Name	Access	Description	Fields
0x00	FIFO_WR	Write-Only	Write Data address	[31:0] Write data
0x08	FIFO_RD	Read-Only	Read Data address	[31:0] Read data
0x10	Status	Read-Only	Status of Mailbox	[31] Empty: • 0 = There is data in the Receive FIFO. • 1 = The Receive FIFO is empty. [30] Full: • 0 = The Send FIFO can receive more data. • 1 = The Send FIFO is full. [29]: Almost Full: • 0 = The Send FIFO level is greater than the WR_Thresholds. • 1 = The Send FIFO level is less or equal to the WR_Thresholds. [28]: Almost Empty: • 0 = The Receive FIFO level is less or equal to the RD_Thresholds. • 1 = The Receive FIFO level is greater than the RD_Thresholds.
0x14	Error	Read-Only	Error Flags. A read clears the status.	[31] Empty Error: Asserts when the processor tries to read the Empty FIFO. The read is ignored. [30] Full Error: Assert when the processor tries to write the Full FIFO. The write is ignored.
0x18	WR_Thresholds	Read-Write	Threshold to raise the write interrupt.	[31:0] Write Thresholds: Thresholds for the Send FIFO
0x1C	RD_Thresholds	Read-Write	Threshold to raise the read interrupt	[31:0] Read Thresholds: Thresholds for the Receive FIFO
0x20	IRQ_STATUS	Read-W1C	Interrupt status	[31]: Write Threshold Interrupt: The Send FIFO level is greater than WR_Threshold. [30]: Read Threshold Interrupt: The Receive FIFO level is greater than or equal to RD_Threshold. [29]: Error Interrupt: The mailbox IP encounters a read or write error.
0x24	IRQ_ENABLE	Read-Write	Enable each type of interrupts	[31]: Write Threshold Interrupt Enable [30]: Read Threshold Interrupt Enable [29]: Error Interrupt Enable
0x28	IRQ_PENDING	Read-only	Current pending interrupts.	[31]: Write Threshold Interrupt Pending [30]: Read Threshold Interrupt Pending [29]: Error Interrupt Pending

2.3. Signal Description

Table 2.2 to Table 2.5 list the ports of the soft IP in different categories.

2.3.1. Clock and Reset

Table 2.2. Clock and Reset Ports

Name	Direction	Width	Description
port0_clk	In	1	Mailbox port 0 system clock.
port1_clk	In	1	Mailbox port 1 system clock.
port0_rst_n	In	1	Port 0 global reset, active low.
port1_rst_n	In	1	Port 1 global reset, active low.

2.3.2. AXI-Lite Interface

Table 2.3. AXI-Lite Subordinate Ports

Name	Direction	Width	Group	Description
AWREADY_Sx	Out	1	AXI4 Mandatory Write Address Channel	—
AWVALID_Sx	In	1		—
AWADDR_Sx	In	32		—
AWPROT_Sx	In	3		Not implemented.
WREADY_Sx	Out	1	AXI4 Mandatory Write Data Channel	—
WVALID_Sx	In	1		—
WDATA_Sx	In	32		—
WSTRB_Sx	In	4		—
BVALID_Sx	Out	1	AXI4 Mandatory Write Response Channel	—
BRESP_Sx	Out	2		b'00: OKAY[1] b'10: SLVERR[1] b'11: DECERR[1]
BREADY_Sx	In	1		—
ARVALID_Sx	In	1	AXI4 Mandatory Read Address Channel	—
ARREADY_Sx	Out	1		—
ARADDR_Sx	In	32		—
RDATA_Sx	Out	32		—
RRESP_Sx	Out	2		b'00: OKAY[1] b'10: SLVERR[1] b'11: DECERR[1]
RVALID_Sx	Out	1		—
RREADY_Sx	In	1		—

Note:

1. Refer to the AMBA AXI Protocol Specification for more detailed descriptions of these responses.
2. Sx represents the subordinate number from 0 to 7.

2.3.3. AHB-Lite Interface

Table 2.4. AHB-Lite Subordinate Ports

Name	Direction	Width	Description
AHBL_Sx_HADDR	In	32	—
AHBL_Sx_HWRITE	In	1	—
AHBL_Sx_HSIZE	In	3	—
AHBL_Sx_HPROT	In	4	Not implemented.
AHBL_Sx_HTRANS	In	2	—
AHBL_Sx_HBURST	In	3	Not implemented.
AHBL_Sx_HMASTLOCK	In	1	Not implemented.
AHBL_Sx_HWDATA	In	32	—
AHBL_Sx_HRDATA	Out	32	—
AHBL_Sx_HREADY	Out	1	—
AHBL_Sx_HRESP	Out	1	—

Note:

1. Refer to the AMBA3 AHBL Protocol Specification for more detailed descriptions of these responses.
2. Sx represents the subordinate number from 0 to 7.

2.3.4. Interrupt Interface

Table 2.5. Interrupt Ports

Name	Type	Width	Description
Irq0	Out	1	Interrupt for Port0.
Irq1	Out	1	Interrupt for Port1.

2.4. Attribute Summary

The configurable attributes of the Mailbox IP are listed and described in Table 2.6 and Table 2.7.

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

Table 2.6. Configurable Attributes

Attribute	Selectable Values	Default	Dependency on Other Attributes
Mailbox FIFO Depth	16 to 8192 four-byte words	16	—
Mailbox FIFO Type	EBR, Distributed-RAM	EBR	—
Enable Output Register	Checked, Unchecked	Unchecked	—
Write Threshold Reset Value	1 to (FIFO Depth – 1)	15	—
Read Threshold Reset Value	1 to (FIFO Depth – 1)	1	—
Interface Type	AXI-L, AHB-L	AXI-L	—

Table 2.7. Attributes Description

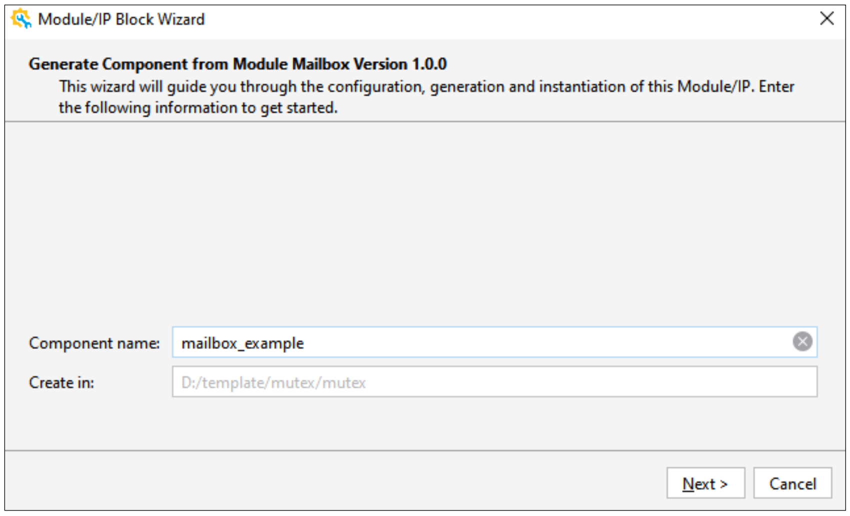
Attribute	Description
Mailbox FIFO Depth	Mailbox FIFO depth
Mailbox FIFO Type	Mailbox FIFO implementation type
Enable Output Register	FIFO output register
Write Threshold Reset Value	FIFO WR_Threshold reset value
Read Threshold Reset Value	FIFO RD_Threshold reset value
Interface Type	Memory-mapped register interface type

3. Mailbox IP Generation

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

To generate the Mailbox IP module:

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



Generate Component from Module Mailbox Version 1.0.0

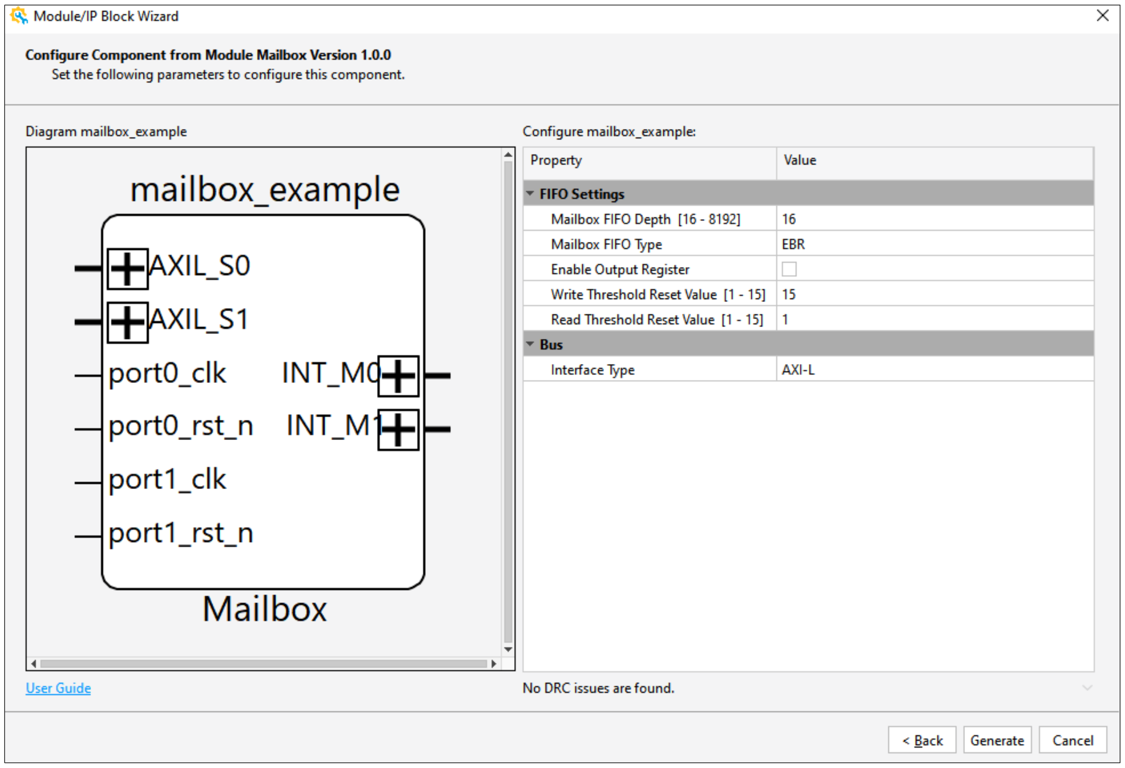
This wizard will guide you through the configuration, generation and instantiation of this Module/IP. Enter the following information to get started.

Component name: mailbox_example

Create in: D:/template/mutex/mutex

Figure 3.1. Entering Component Name

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



Configure Component from Module Mailbox Version 1.0.0

Set the following parameters to configure this component.

Diagram mailbox_example

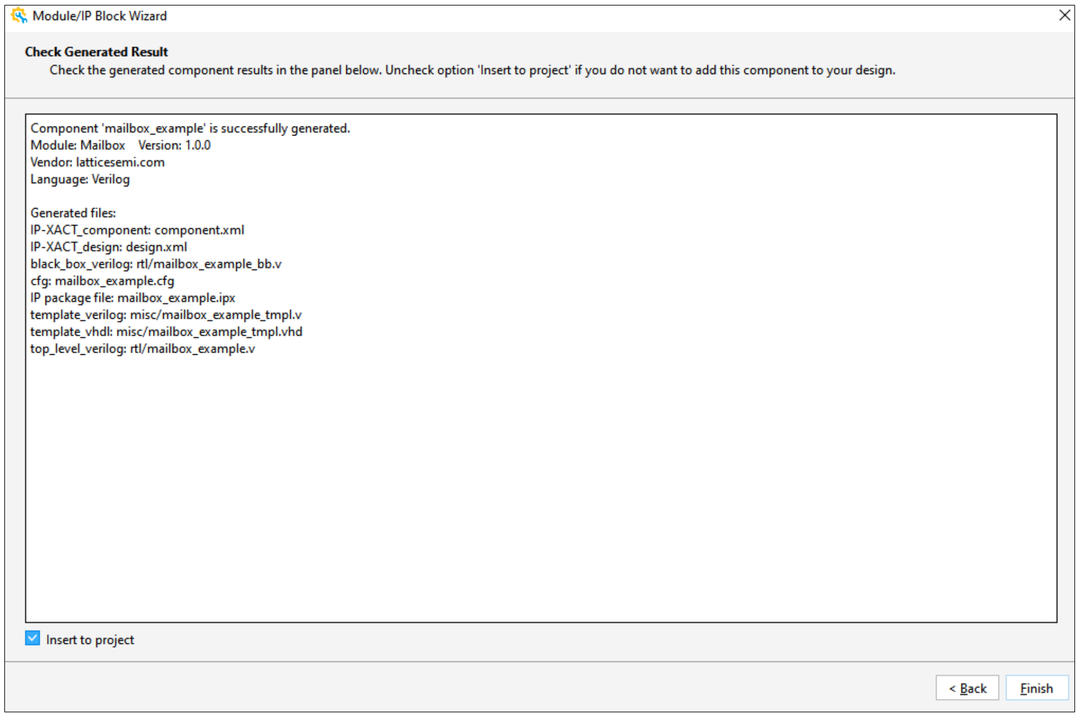
Configure mailbox_example:

Property	Value
FIFO Settings	
Mailbox FIFO Depth [16 - 8192]	16
Mailbox FIFO Type	EBR
Enable Output Register	☐
Write Threshold Reset Value [1 - 15]	15
Read Threshold Reset Value [1 - 15]	1
Bus	
Interface Type	AXI-L

No DRC issues are found.

Figure 3.2. Configuring Parameters

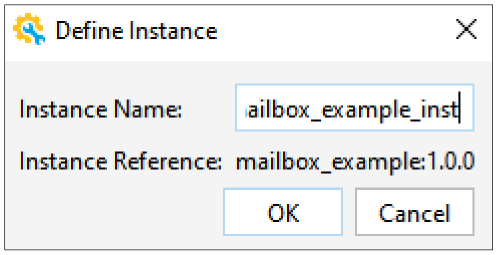
4. Verify the information. Click **Finish** (Figure 3.3).


Module/IP Block Wizard
Check Generated Result
Check the generated component results in the panel below. Uncheck option 'Insert to project' if you do not want to add this component to your design.

```
Component 'mailbox_example' is successfully generated.
Module: Mailbox    Version: 1.0.0
Vendor: latticesemi.com
Language: Verilog

Generated files:
IP-XACT_component: component.xml
IP-XACT_design: design.xml
black_box_verilog: rtl/mailbox_example_bb.v
cfg: mailbox_example.cfg
IP package file: mailbox_example.ipx
template_verilog: misc/mailbox_example_tmpl.v
template_vhdl: misc/mailbox_example_tmpl.vhd
top_level_verilog: rtl/mailbox_example.v
```

Figure 3.3. Verifying Results

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



Instance Name: ailbox_example_inst

Instance Reference: mailbox_example:1.0.0

Figure 3.4. Specifying Instance Name

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

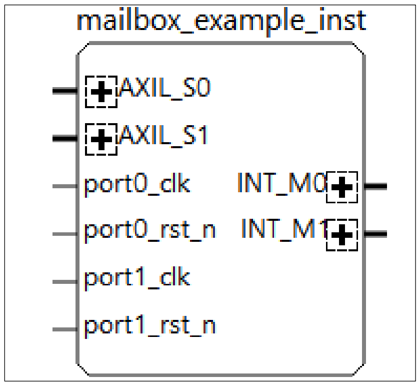


Figure 3.5. Generated Instance

4. Example Design

A system design that incorporates the Mailbox IP can be generated using the RISC-V MC Multi-Processor Project template available in Lattice Propel 2025.2. This design is referred to as the Example Design throughout this guide. The Example Design is documented in the accompanying README file and validated through a C code test sequence. Additional notes and explanations are provided as comments within the test code to help you understand the implementation and behavior of the Mailbox IP.

You can find the README.txt file at the following location:

<Propel Installation Path>/templates/MULTI_PROCESSOR_Template01/Readme.txt
This example design enables you to compile and test the Mailbox IP functionality on the following Lattice evaluation board:

Avant-E Evaluation Board

4.1. Supported Configuration

In the example design, the following IP configuration is applied to the Mailbox IP.

Note: In Table 4.1, "√" refers to a checked option for Mailbox IP in the example design and "×" refers to an unchecked option or a non-applicable option in the Mailbox IP.

Table 4.1. Mailbox IP Configuration Used in Example Design

Mailbox IP GUI Parameter	Mailbox IP Configuration
FIFO Settings	
Mailbox FIFO Depth [16–8192]	32
Mailbox FIFO Type	EBR
Enable Output Register	mailbox01_inst: √ mailbox02_inst: × mailbox03_inst: × mailbox04_inst: × mailbox05_inst: ×
Write Threshold Reset Value [1–31]	15
Read Threshold Reset Value [1–31]	1
Bus	
Interface Type	AHB-Lite

4.2. Overview of the Example Design and Features

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

- Processor
  - CPU0 (Main) - RISC-V MC with PIC and timer
  - CPU1-5 – RISC-V MC with PIC
- AHBL to APB Bridge
- AHBL Interconnect
- APB Interconnect
- GPIO
- Mailbox
- Mutex
- IOPMP
- UART
- System Memory

This example design demonstrates the functionality of the Mailbox IP in a multi-core system.

The system comprises six CPUs, from CPU0 to CPU5. CPU0 acts as the system controller, responsible for distributing tasks to CPU1–CPU5. These worker CPUs receive their assigned tasks through the Mailbox, execute the computations independently, and then send their results and status back to CPU0 using the same Mailbox mechanism.

For status output, all CPUs share a common UART interface. To prevent concurrent access to the UART, a Mutex is employed. This ensures that only one CPU can transmit data at a time, while others wait until the Mutex is released, maintaining data integrity and avoiding conflicts.

Figure 4.1 shows the block diagram of the RISC-V MC Multi-Processor project with the Mailbox IP.

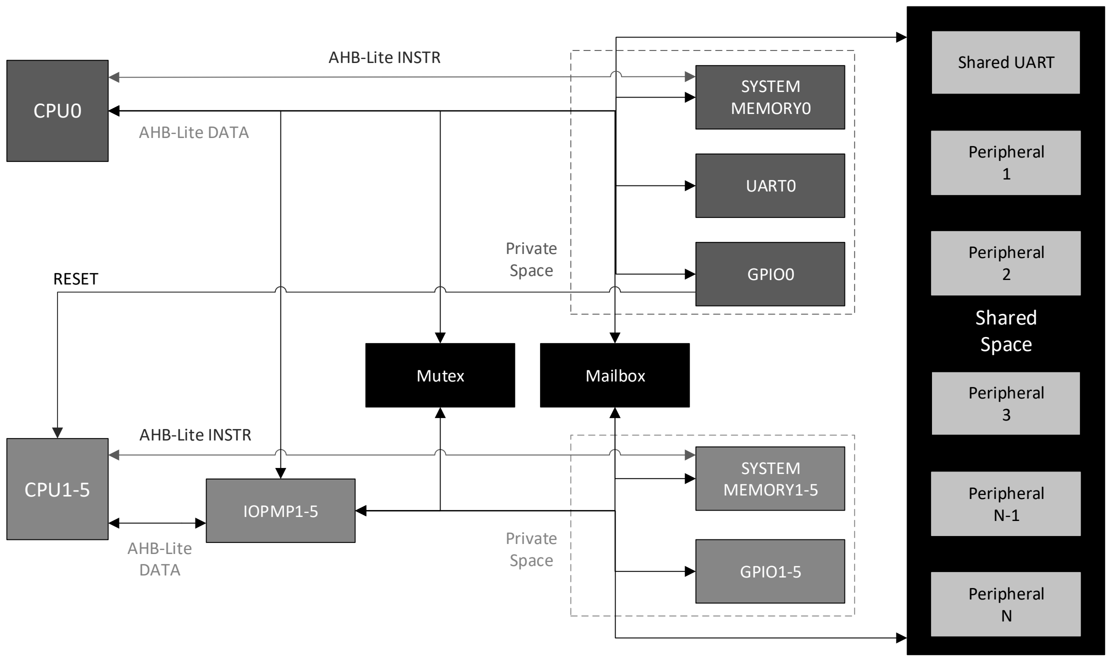


Figure 4.1. RISC-V MC Multi-Processor Project with Mailbox IP Block Diagram

Note: The block diagram does not cover all components of the example design and serves only as a guideline.

4.3. Generating the Example Design

The Lattice Propel Builder and Lattice Propel SDK are used to create an embedded system that includes an SoC design with an embedded processor and system software. This section outlines the procedure for generating an example design that incorporates the Mailbox IP. The hardware design is created using Lattice Propel Builder, while the software workspace is set up and managed within Lattice Propel SDK to develop, build, and deploy the corresponding firmware for each processor core.

To generate the example design:

1. Launch Lattice Propel Builder.

   Open the application, then navigate to **File** > **New SoC Design**, as shown in Figure 4.2.

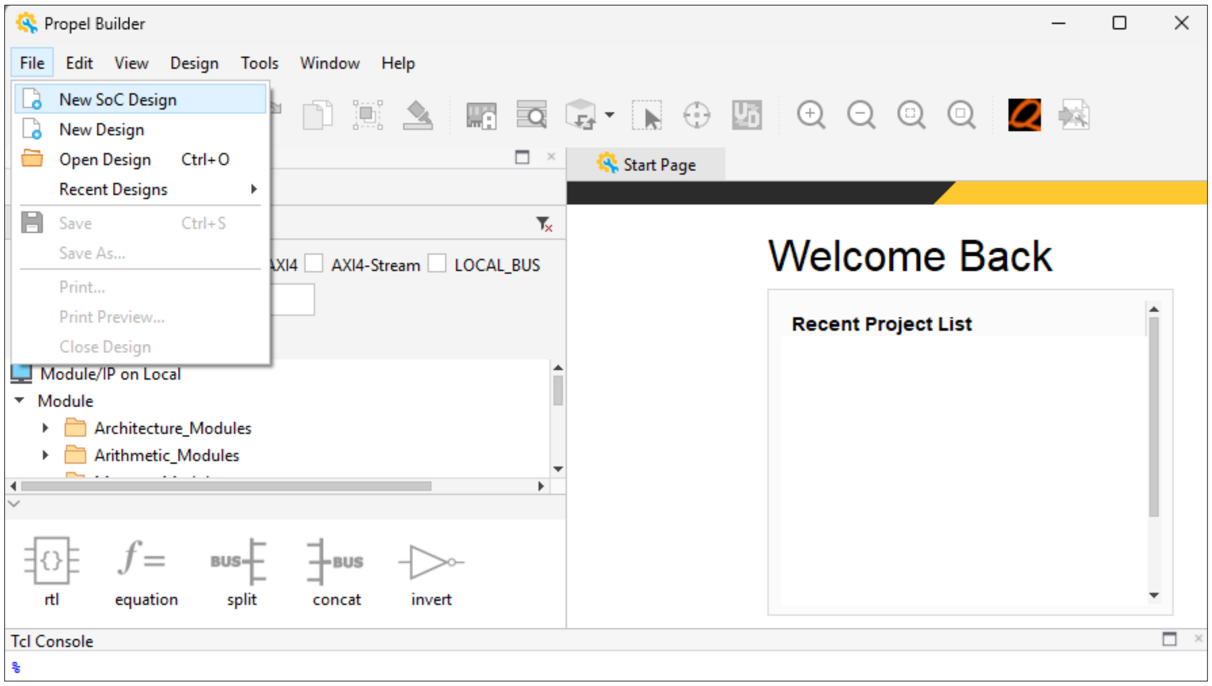


Figure 4.2. Lattice Propel Builder – Create New SoC Design

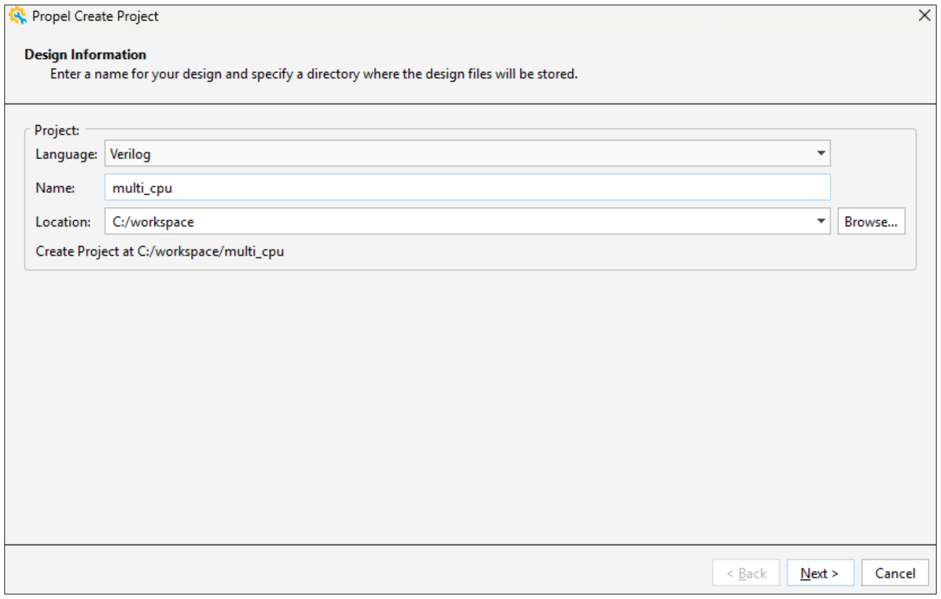
2. Enter the project Information.

   In the **Create Project** > **Design Information** window:

   a. Enter a name for your project.

   b. Browse to select a directory for the project location.

   c. Click **Next** to proceed (Figure 4.3).



Design Information

Enter a name for your design and specify a directory where the design files will be stored.

Project:

Language: Verilog

Name: multi_cpu

Location: C:/workspace

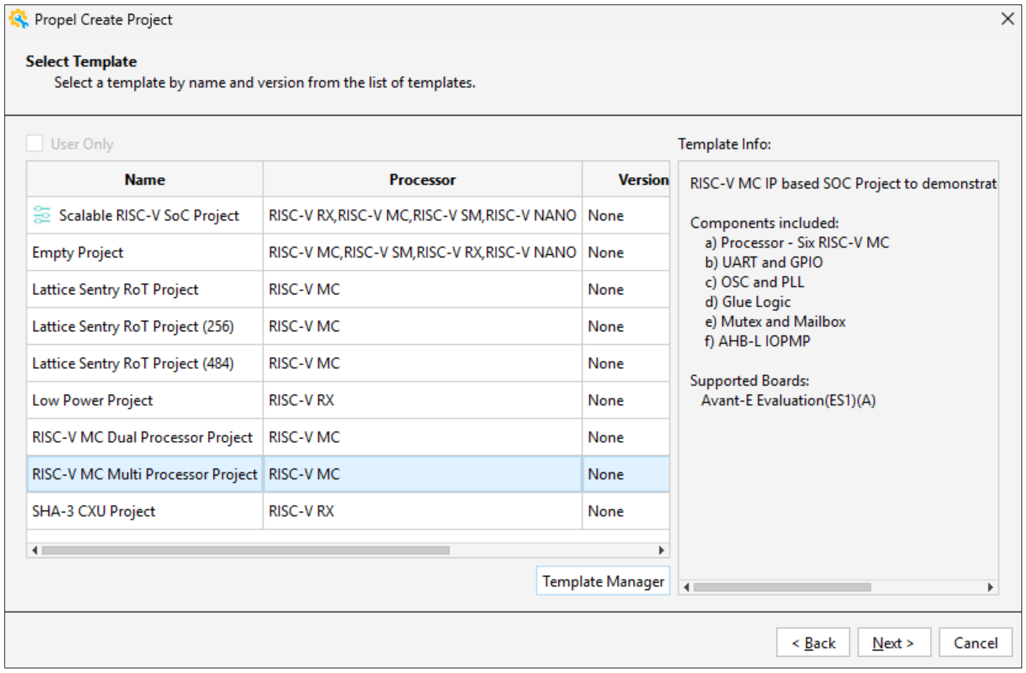
Create Project at C:/workspace/multi_cpu

Figure 4.3. Create Project – Design Information

3. Select a Project Template.

   In the **Create Project > Select Template** window:

a. Choose **RISC-V MC Multi-Processor Project** from the list of available templates.
b. Click **Next** to continue (Figure 4.4).



Propel Create Project

Select Template
Select a template by name and version from the list of templates.

User Only

Name	Processor	Version
Scalable RISC-V SoC Project	RISC-V RX,RISC-V MC,RISC-V SM,RISC-V NANO	None
Empty Project	RISC-V MC,RISC-V SM,RISC-V RX,RISC-V NANO	None
Lattice Sentry RoT Project	RISC-V MC	None
Lattice Sentry RoT Project (256)	RISC-V MC	None
Lattice Sentry RoT Project (484)	RISC-V MC	None
Low Power Project	RISC-V RX	None
RISC-V MC Dual Processor Project	RISC-V MC	None
RISC-V MC Multi Processor Project	RISC-V MC	None
SHA-3 CXU Project	RISC-V RX	None

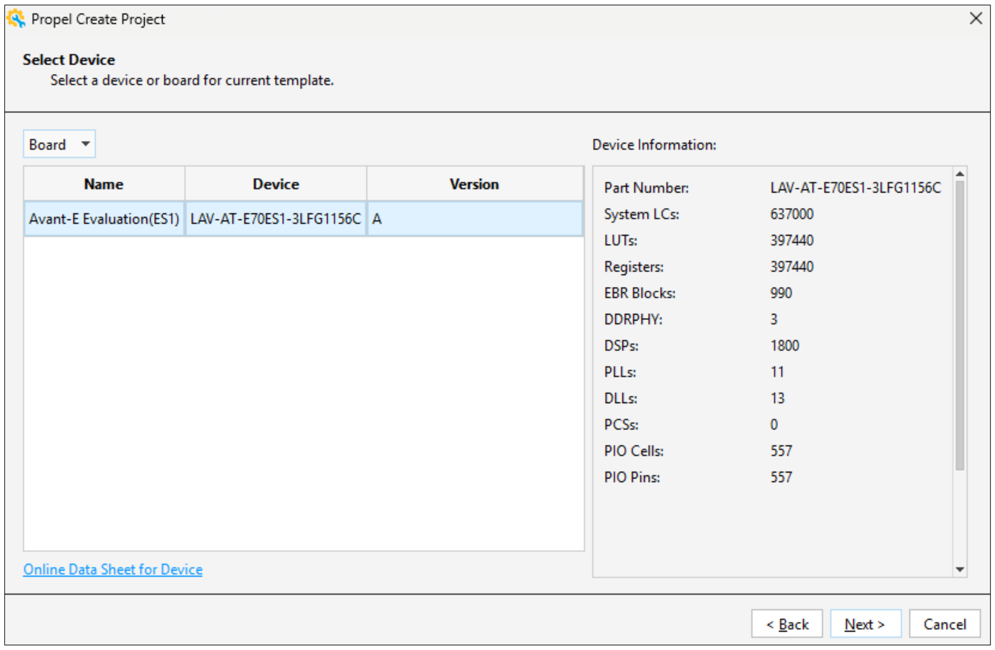
Template Info:
RISC-V MC IP based SOC Project to demonstrat

Components included:
a) Processor - Six RISC-V MC
b) UART and GPIO
c) OSC and PLL
d) Glue Logic
e) Mutex and Mailbox
f) AHB-L IOPMP

Supported Boards:
Avant-E Evaluation(ES1)(A)

Figure 4.4. Select Template – RISC-V MC Multi-Processor Project

4. Select the Target Device.
   In the **Create Project > Select Device** window:
   a. Select **Avant-E Evaluation (ES1)** as the target device.
   b. Click **Next**, then click **Finish** to complete the project setup (Figure 4.5).



Propel Create Project

Select Device
Select a device or board for current template.

Board

Name	Device	Version
Avant-E Evaluation(ES1)	LAV-AT-E70ES1-3LFG1156C	A

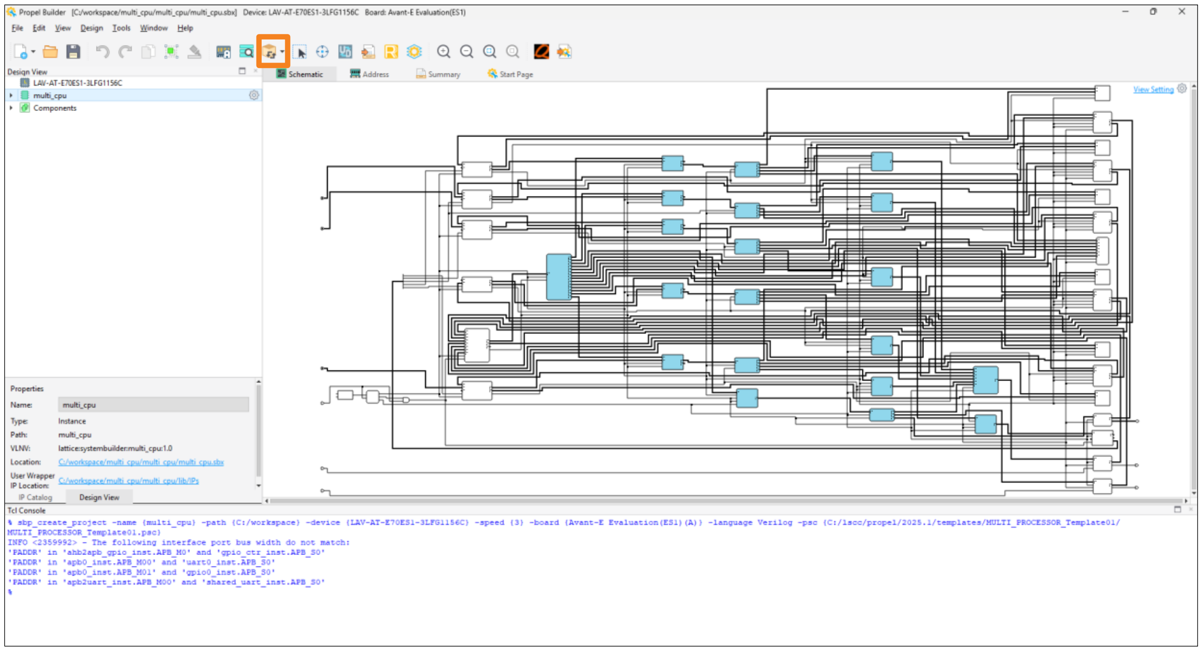
Device Information:

Part Number:	LAV-AT-E70ES1-3LFG1156C
System LCs:	637000
LUTs:	397440
Registers:	397440
EBR Blocks:	990
DDRPHY:	3
DSPs:	1800
PLLs:	11
DLLs:	13
PCSs:	0
PIO Cells:	557
PIO Pins:	557

Figure 4.5. Select Target Device – Avant-E Evaluation (ES1)

5. Generate the SoC design.
   a. Once the project is created, the **Lattice Propel Builder** main window opens.
   b. Click the **Generate** button to build the SoC design (Figure 4.6).

   Note: Warnings related to TIMER_IRQ and INT may appear. These are expected and can be safely ignored.



Properties

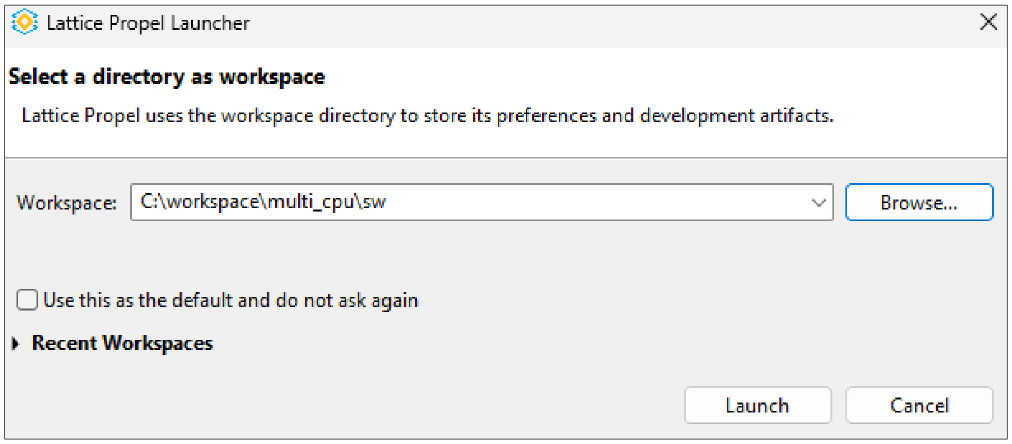
Name:	multi_cpu
Type:	Instance
Path:	multi_cpu
VLNV:	lattice:systembuilder:multi_cpu:1.0
Location:	C:/workspace/multi_cpu/multi_cpu/multi_cpu.sbx
User Wrapper IP Location:	C:/workspace/multi_cpu/multi_cpu/lib/IPs

```
% sbp_create_project -name {multi_cpu} -path {C:/workspace} -device {LAV-AT-E70ES1-3LFG1156C} -speed {3} -board {Avant-E Evaluation(ES1)(A)} -language Verilog -psc {C:/lscc/propel/2025.1/templates/MULTI_PROCESSOR_Template01/
MULTI_PROCESSOR_Template01.psc}
INFO <2359992> - The following interface port bus width do not match:
'PADDR' in 'ahb2apb_gpio_inst.APB_M0' and 'gpio_ctr_inst.APB_S0'
'PADDR' in 'apb0_inst.APB_M00' and 'uart0_inst.APB_S0'
'PADDR' in 'apb0_inst.APB_M01' and 'gpio0_inst.APB_S0'
'PADDR' in 'apb2uart_inst.APB_M00' and 'shared_uart_inst.APB_S0'
%
```

Propel Builder [C:/workspace/multi_cpu/multi_cpu/multi_cpu.sbx] Device: LAV-AT-E70ES1-3LFG1156C Board: Avant-E Evaluation(ES1)
File Edit View Design Tools Window Help
Design View
LAV-AT-E70ES1-3LFG1156C
multi_cpu
Components
Schematic
Address
Summary
Start Page
View Setting
IP Catalog
Design View
Tcl Console

Figure 4.6. Lattice Propel Builder – Generate SoC Design

6. Launch Lattice Propel SDK.
   a. Open Lattice Propel SDK. The Lattice Propel Launcher window appears.
   b. Browse to select a directory for your workspace.
   c. Click **Launch** to open the SDK environment (Figure 4.7).



Lattice Propel Launcher

Select a directory as workspace

Lattice Propel uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\workspace\multi_cpu\sw

☐ Use this as the default and do not ask again

Figure 4.7. Lattice Propel SDK – Select Workspace and Launch

7. Create a New C/C++ Project.
   In Lattice Propel SDK:
   Navigate to **File > New > Lattice C/C++ Project** to begin creating a new software project (Figure 4.8).

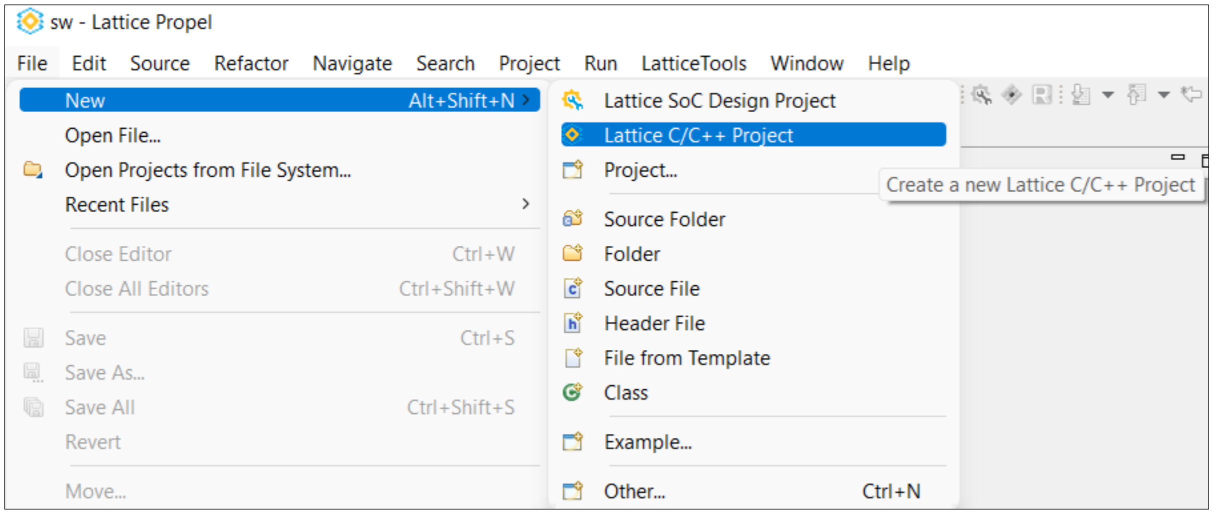
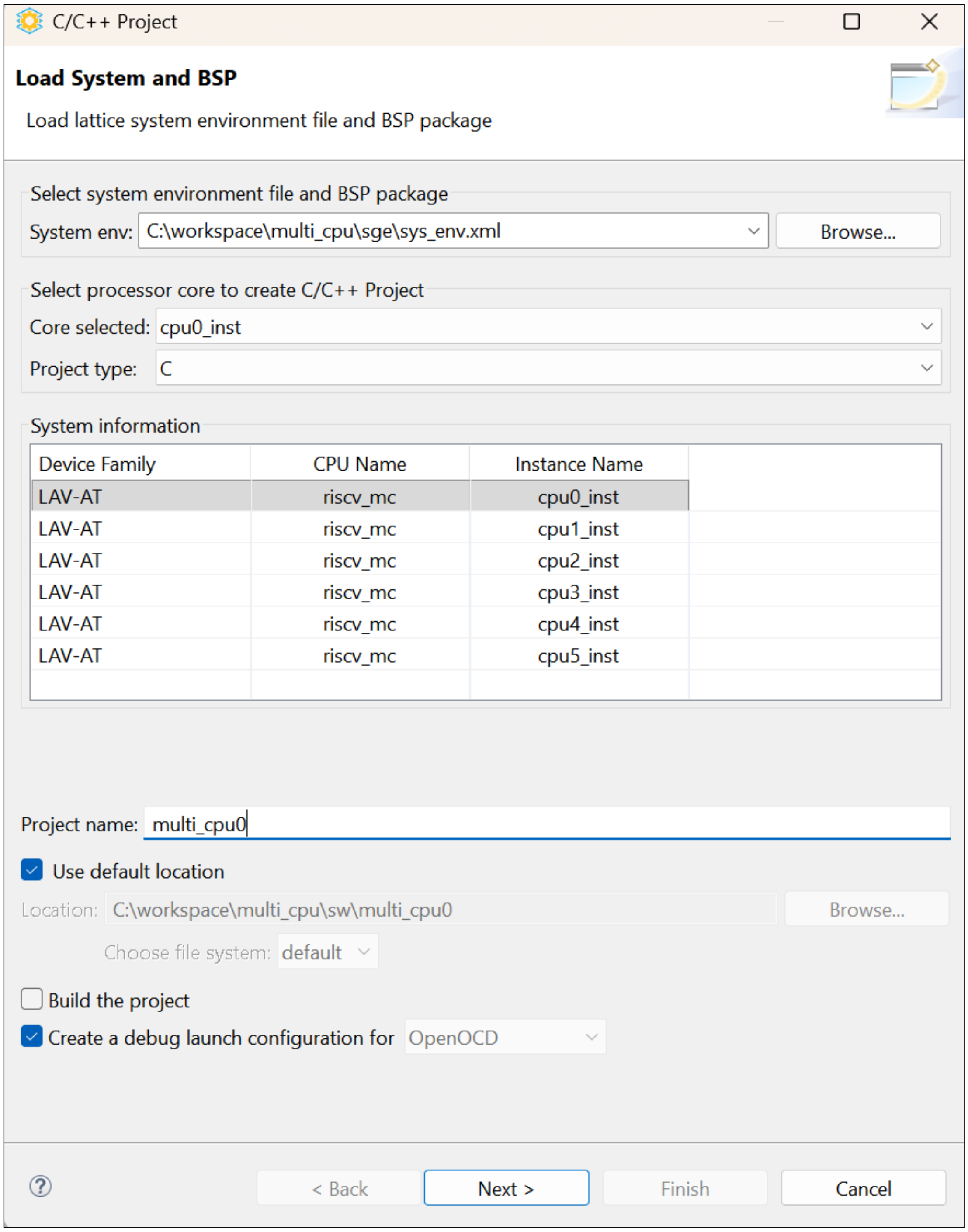


Figure 4.8. Lattice Propel SDK – Create New C/C++ Project

8. Load System and BSP Configuration.
   a. Browse to the sys_env.xml file generated by Lattice Propel Builder, as shown in Figure 4.9.
   b. Select **cpu0_inst** as the target processor.
   c. Enter a name for your project.
   d. Click **Next**. Then, click **Finish** to complete the setup.



Load System and BSP

Load lattice system environment file and BSP package

Select system environment file and BSP package

System env: C:\workspace\multi_cpu\sge\sys_env.xml

Select processor core to create C/C++ Project

Core selected: cpu0_inst

Project type: C

System information

Device Family	CPU Name	Instance Name
LAV-AT	riscv_mc	cpu0_inst
LAV-AT	riscv_mc	cpu1_inst
LAV-AT	riscv_mc	cpu2_inst
LAV-AT	riscv_mc	cpu3_inst
LAV-AT	riscv_mc	cpu4_inst
LAV-AT	riscv_mc	cpu5_inst

Project name: multi_cpu0

☑ Use default location

Location: C:\workspace\multi_cpu\sw\multi_cpu0

Choose file system: default

☐ Build the project

☑ Create a debug launch configuration for OpenOCD

Figure 4.9. Load System and BSP – Select cpu0_inst

9. Repeat steps 7 and 8 for Remaining CPUs.

Repeat **Step 7 and Step 8** for each of the remaining CPUs, from CPU1 to CPU5. Select the corresponding processor instance, such as cpu1_inst, cpu2_inst, and so on during each setup, as shown in Table 4.2.

Table 4.2. Load System and BSP – Select Remaining CPUs

Core Selected	Project Name
cpu0_inst	multi_cpu0
cpu1_inst	multi_cpu1
cpu2_inst	multi_cpu2
cpu3_inst	multi_cpu3
cpu4_inst	multi_cpu4
cpu5_inst	multi_cpu5

10. Build all software projects.
    After all CPU software projects, from CPU0 to CPU5, have been created:
    Navigate to **Project > Build All** to compile all projects simultaneously (Figure 4.10).

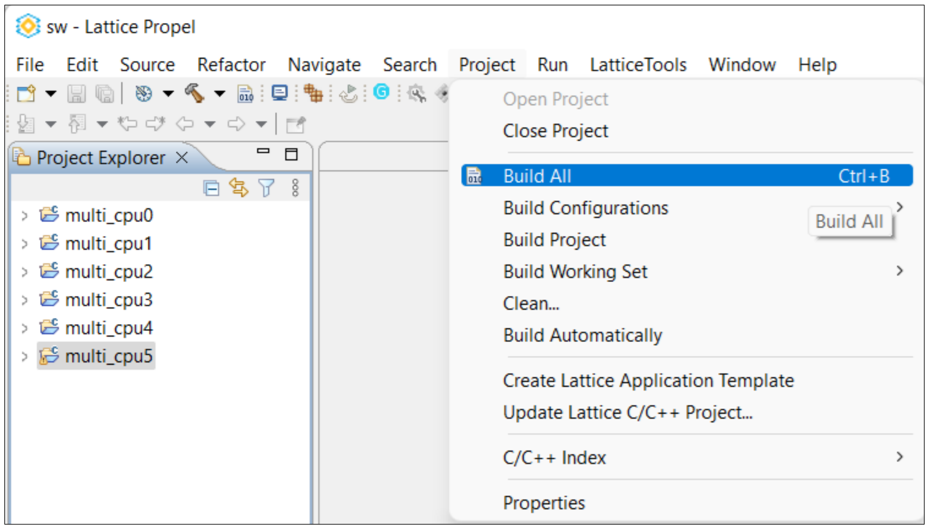
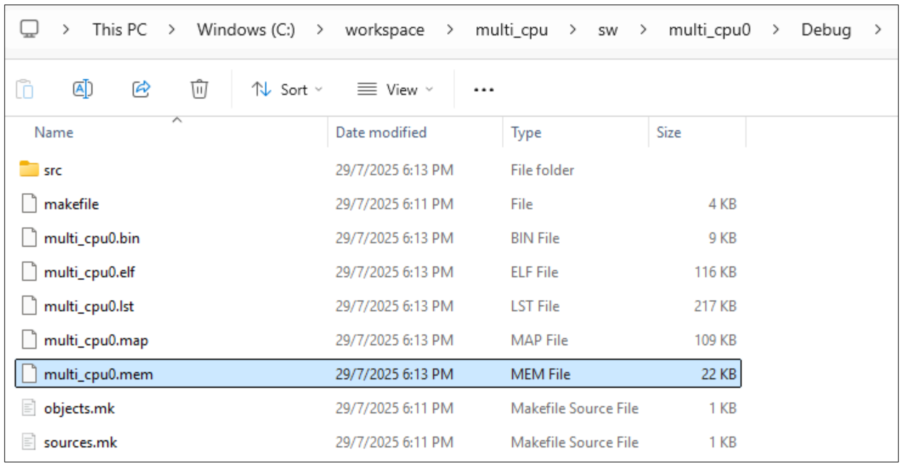


Figure 4.10. Lattice Propel SDK – Build All Projects

11. Verify the .mem file generation.
    Ensure that a .mem file is generated for each project (Figure 4.11). These files should be located in the following directory: C:\workspace\multi_cpu\sw\multi_cpu*\Debug
    Note: * represents CPU indices 0 through 5.

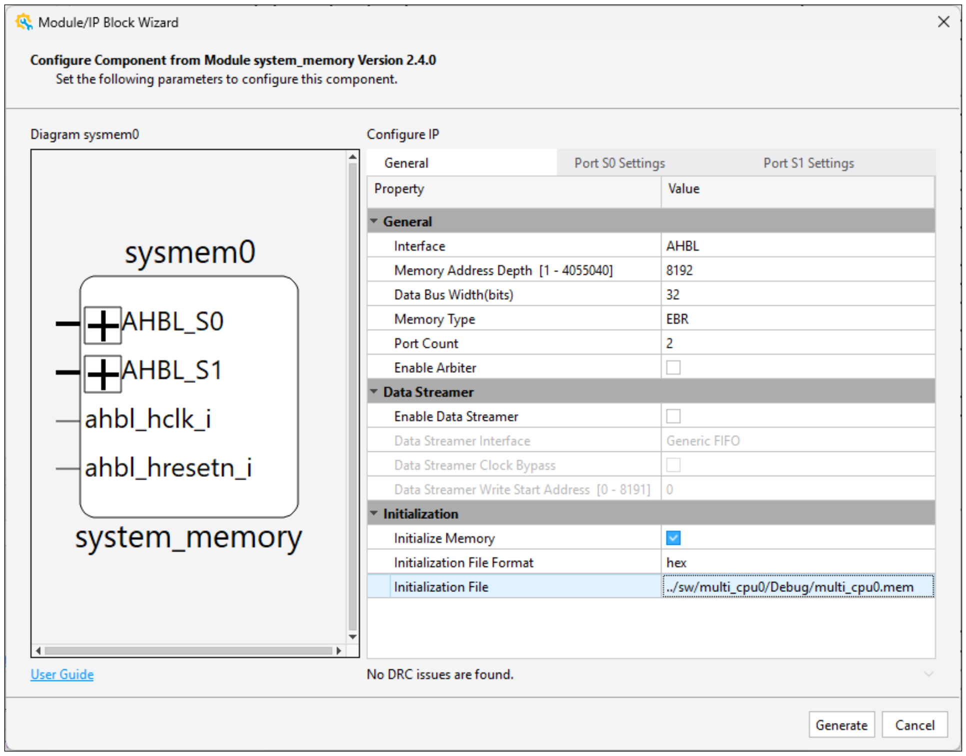

This PC > Windows (C:) > workspace > multi_cpu > sw > multi_cpu0 > Debug >
Sort
View

Name	Date modified	Type	Size
src	29/7/2025 6:13 PM	File folder	
makefile	29/7/2025 6:11 PM	File	4 KB
multi_cpu0.bin	29/7/2025 6:13 PM	BIN File	9 KB
multi_cpu0.elf	29/7/2025 6:13 PM	ELF File	116 KB
multi_cpu0.lst	29/7/2025 6:13 PM	LST File	217 KB
multi_cpu0.map	29/7/2025 6:13 PM	MAP File	109 KB
multi_cpu0.mem	29/7/2025 6:13 PM	MEM File	22 KB
objects.mk	29/7/2025 6:11 PM	Makefile Source File	1 KB
sources.mk	29/7/2025 6:11 PM	Makefile Source File	1 KB

Figure 4.11. File Explorer – .mem Files for Each CPU Project

12. Initialize System Memory in Lattice Propel Builder (Figure 4.12).

    In Lattice Propel Builder, initialize each system memory block (sysmem)* with its corresponding .mem file, as shown in Table 4.3.


Module/IP Block Wizard

Configure Component from Module system_memory Version 2.4.0
Set the following parameters to configure this component.

Diagram sysmem0

Configure IP

General | Port S0 Settings | Port S1 Settings

Property	Value
General	
Interface	AHBL
Memory Address Depth [1 - 4055040]	8192
Data Bus Width(bits)	32
Memory Type	EBR
Port Count	2
Enable Arbiter	☐
Data Streamer	
Enable Data Streamer	☐
Data Streamer Interface	Generic FIFO
Data Streamer Clock Bypass	☐
Data Streamer Write Start Address [0 - 8191]	0
Initialization	
Initialize Memory	✓
Initialization File Format	hex
Initialization File	../sw/multi_cpu0/Debug/multi_cpu0.mem

User Guide

No DRC issues are found.

Generate Cancel

Figure 4.12. Lattice Propel Builder – Initialize System Memory with .mem Files

Table 4.3. Initialize System Memory with .mem Files

System Memory Instance Name	.mem File Name
sysmem0_inst	multi_cpu0.mem
sysmem1_inst	multi_cpu1.mem
sysmem2_inst	multi_cpu2.mem
sysmem3_inst	multi_cpu3.mem
sysmem4_inst	multi_cpu4.mem
sysmem5_inst	multi_cpu5.mem

13. Regenerate the SoC design.

    After the memory initialization:

    Click the **Generate** button again in Lattice Propel Builder to update the design with the new memory contents (Figure 4.13).

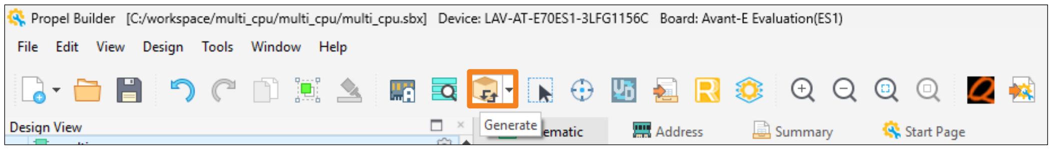

Propel Builder [C:/workspace/multi_cpu/multi_cpu/multi_cpu.sbx] Device: LAV-AT-E70ES1-3LFG1156C Board: Avant-E Evaluation(ES1)

Figure 4.13. Lattice Propel Builder – Regenerate Design After Memory Initialization

14. Launch the Lattice Radiant software and compile the design.

Within Lattice Propel Builder:

a. Launch the Lattice Radiant software.

b. Compile the design to complete the hardware build process (Figure 4.14).

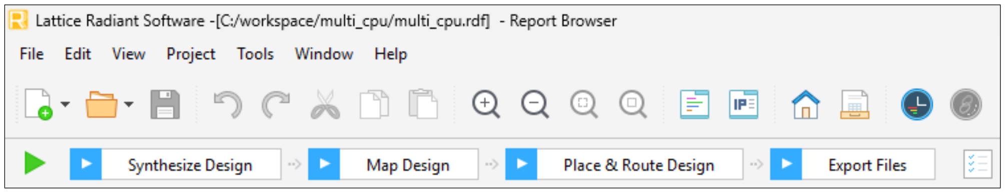


Figure 4.14. Launch Lattice Radiant – Compile Design

4.4. Running the Example Design

To view the output, set up a UART terminal as described in the instructions below. The bitstream file generated in the Generating the Example Design section is downloaded to the Avant-E Evaluation Board using the Lattice Radiant Programmer.

4.4.1. Setting Up the UART Terminal

The software code in this example design displays messages on a terminal through the UART interface. To view the output, follow steps below to set up a UART terminal:

1. Launch Tera Term, or use Serial Terminal in SDK, or PuTTY.
2. Go to **File > New Connection.**
3. Select **Serial.**
4. Choose the correct COM port, which is usually the higher number.
5. Click **OK.**
6. Go to **Setup > Serial Port** and configure:
   - Baud Rate: 115200
   - Data: 8 bits
   - Parity: None
   - Stop Bits: 1
   - Flow Control: None
   - Transmit Delay**:** 0 msec/char; 0 msec/line
7. Click **New Setting** to complete the setup process.

4.4.2. Programming the FPGA Bitstream

To program the bitstream into the device, follow these steps:

1. Launch Radiant Programmer.

   Open the Lattice Radiant software, then go to **Tools > Programmer** (Figure 4.15).

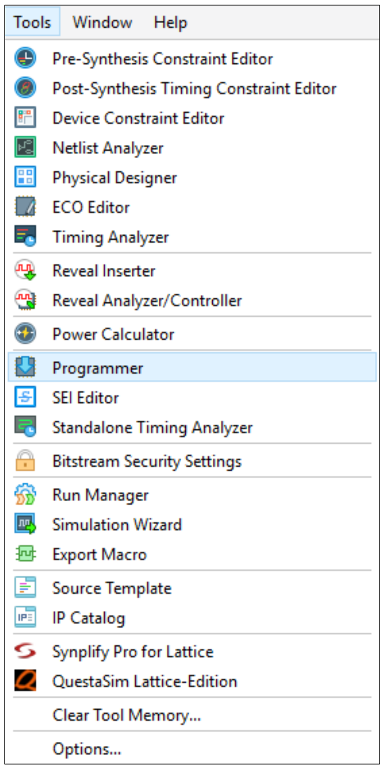


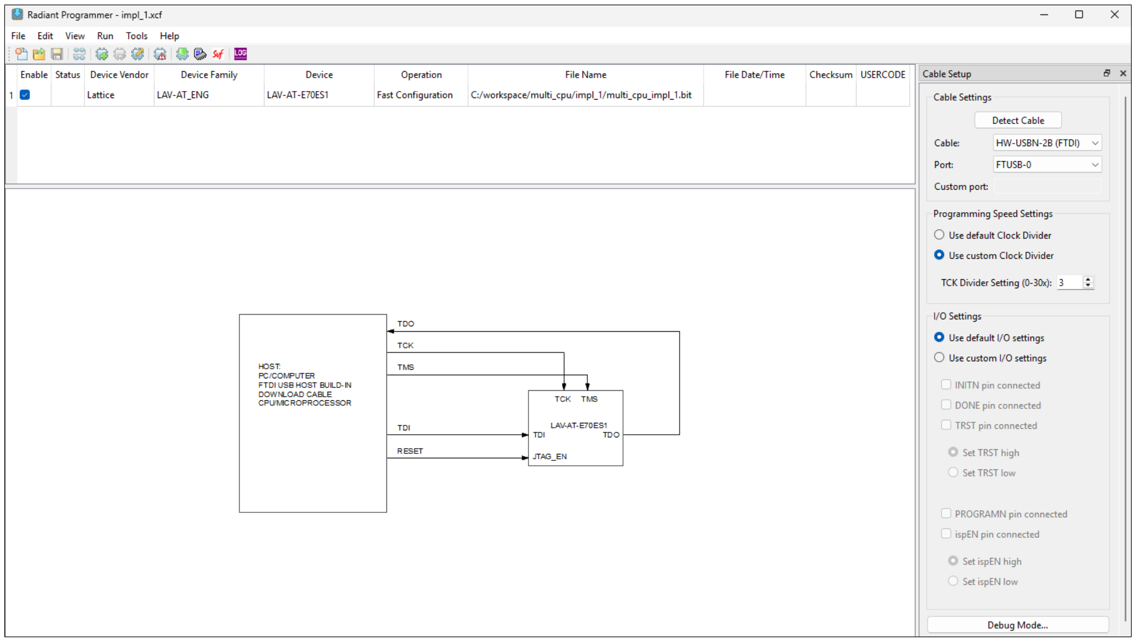
Figure 4.15. Launching Radiant Programmer

2. Configure the programmer settings.

   In the Lattice Radiant Programmer window:

   a. Ensure the **Device Family**, **Device**, and **Operation** match the configuration shown in Figure 4.16.

   b. Select the .bit file generated from previous steps using the Lattice Radiant software.



Radiant Programmer - impl_1.xcf

	Enable	Status	Device Vendor	Device Family	Device	Operation	File Name	File Date/Time	Checksum	USERCODE
1	✓		Lattice	LAV-AT_ENG	LAV-AT-E70ES1	Fast Configuration	C:/workspace/multi_cpu/impl_1/multi_cpu_impl_1.bit			

Cable Setup

Cable Settings

Cable: HW-USBN-2B (FTDI)

Port: FTUSB-0

Custom port:

Programming Speed Settings

Use default Clock Divider [✓ Use custom Clock Divider]

TCK Divider Setting (0-30x): 3

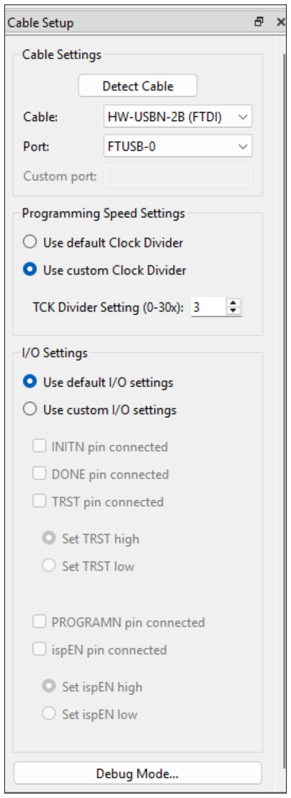
I/O Settings

[✓ Use default I/O settings] Use custom I/O settings

INITN pin connected
DONE pin connected
TRST pin connected
Set TRST high
Set TRST low
PROGRAMN pin connected
ispEN pin connected
Set ispEN high
Set ispEN low

Figure 4.16. Lattice Radiant Programmer – Device and Bitstream Configuration

3. Detect the programming cable.
   In the **Cable Setup** window, click **Detect Cable** (Figure 4.17).


Cable Setup

Cable Settings

Detect Cable

Cable: HW-USBN-2B (FTDI)

Port: FTUSB-0

Custom port:

Programming Speed Settings

( ) Use default Clock Divider

(●) Use custom Clock Divider

TCK Divider Setting (0-30x): 3

I/O Settings

(●) Use default I/O settings

( ) Use custom I/O settings

[ ] INITN pin connected

[ ] DONE pin connected

[ ] TRST pin connected

(●) Set TRST high

( ) Set TRST low

[ ] PROGRAMN pin connected

[ ] ispEN pin connected

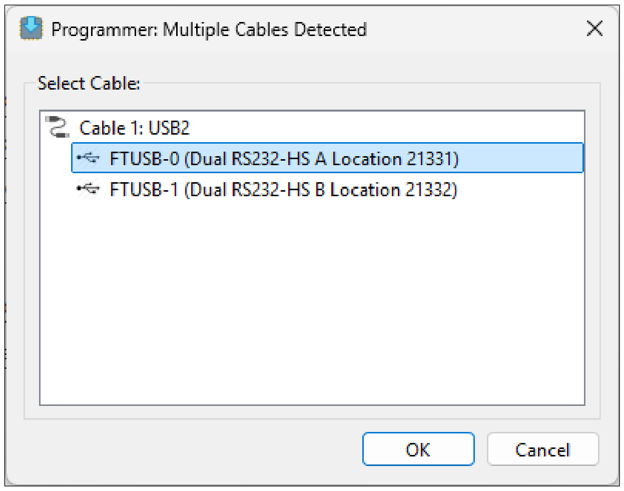
(●) Set ispEN high

( ) Set ispEN low

Debug Mode...

Figure 4.17. Cable Setup – Detecting Cable

4. Select the programming cable.
   In the **Select Cable** window:
   a. Choose **FTUSB-0** from the list.
   b. Click **OK** to confirm (Figure 4.18).


Programmer: Multiple Cables Detected

Select Cable:

Cable 1: USB2

FTUSB-0 (Dual RS232-HS A Location 21331)

FTUSB-1 (Dual RS232-HS B Location 21332)

OK Cancel

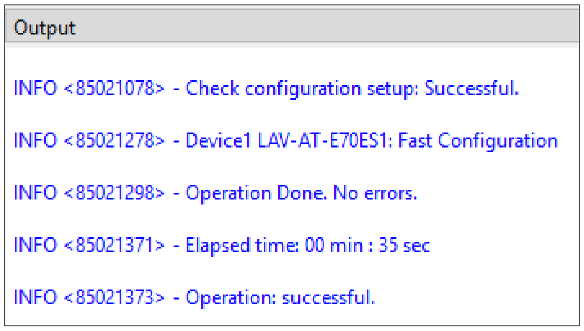
Figure 4.18. Select Cable – FTUSB-0 Option

5. Program the device.

   In the Lattice Radiant Programmer main interface:

   a. Go to **Run** > **Program Device**.

   b. The **Output Console** displays a message indicating that the operation is successful (Figure 4.19).



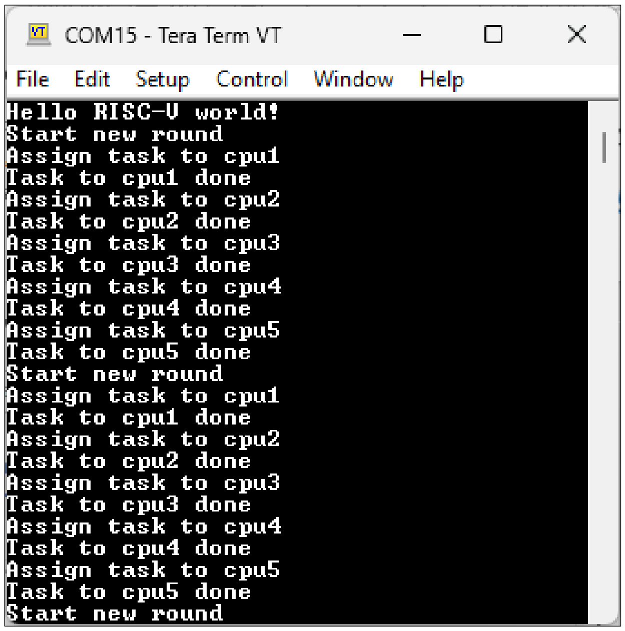
```
INFO <85021078> - Check configuration setup: Successful.

INFO <85021278> - Device1 LAV-AT-E70ES1: Fast Configuration

INFO <85021298> - Operation Done. No errors.

INFO <85021371> - Elapsed time: 00 min : 35 sec

INFO <85021373> - Operation: successful.
```

Figure 4.19. Output Console – Programming the FPGA Bitstream

4.4.3. Verifying Results

Verify the UART terminal output.

After programming the FPGA and running the design, check the UART terminal, such as Tera Term, to ensure the printed messages match the expected output, as shown in Figure 4.20.



```
Hello RISC-V world!
Start new round
Assign task to cpu1
Task to cpu1 done
Assign task to cpu2
Task to cpu2 done
Assign task to cpu3
Task to cpu3 done
Assign task to cpu4
Task to cpu4 done
Assign task to cpu5
Task to cpu5 done
Start new round
Assign task to cpu1
Task to cpu1 done
Assign task to cpu2
Task to cpu2 done
Assign task to cpu3
Task to cpu3 done
Assign task to cpu4
Task to cpu4 done
Assign task to cpu5
Task to cpu5 done
Start new round
```

Figure 4.20. UART Terminal – Expected Output Messages

Appendix A. Resource Utilization

Table A.1. Resource Utilization in LFCPNX-100 Device

Configuration	LUTs	Registers	sysMEM EBRs
Interface = AHB-Lite, FIFO Depth = 16, FIFO Type = EBR	360	195	2
Interface = AHB-Lite, FIFO Depth = 16, FIFO Type = Distributed-RAM	456	259	0
Interface = AHB-Lite, FIFO Depth = 8192, FIFO Type = EBR	759	440	32

Table A.2. Resource Utilization in LAV-AT-E70 Device

Configuration	LUTs	Registers	sysMEM EBRs
Interface = AHB-Lite, FIFO Depth = 16, FIFO Type = EBR	341	183	2
Interface = AHB-Lite, FIFO Depth = 16, FIFO Type = Distributed-RAM	441	247	0
Interface = AHB-Lite, FIFO Depth = 8192, FIFO Type = EBR	728	404	16

References

- Lattice Propel Builder 2025.2 User Guide (FPGA-UG-02243)
- Lattice Memory Mapped Interface and Lattice Interrupt Interface User Guide (FPGA-UG-02039)
- AMBA AXI and ACE Protocol Specification
- AMBA 3 AHB-Lite Protocol Specification

For more information, refer to:

- Lattice Propel web page
- Lattice Certus-N2 Family Devices 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
- 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 v1.0.0, December 2025

Section	Change Summary
All	Production release.

www.latticesemi.com